# Vehicle Egomotion Estimation Using Computer Vision

by

Robert Martin Panish

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2008



Author .....
Department of Aeronautics and Astronautics
September 1, 2008

Certified by .....
Prof. Emilio Frazzoli
Associate Professor
Thesis Supervisor

Accepted by .....
Prof. David L. Darmofal
Associate Department Head
Chair, Committee on Graduate Students

**Vehicle Egomotion Estimation Using Computer Vision**

by

Robert Martin Panish

Submitted to the Department of Aeronautics and Astronautics
on September 1, 2008, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

## Abstract

A vision based navigation filter is developed for application on UAVs and tested in simulation. This filter is meant to allow the UAV to navigate in GPS-denied environments using measurements from a suite of cameras. The extended Kalman filter integrates measurements from multiple non-overlapping cameras as well as an IMU and occasional GPS. Simulations are conducted to evaluate the performance of the filter in a variety of flight regimes as well as to assess the value of using multiple cameras.

Simulations demonstrate the value of using multiple cameras for egomotion estimation. Multiple non-overlapping cameras are useful for resolving motion in an unobservable direction that manifests as an ambiguity between translation and rotation. Additionally, multiple cameras are extremely useful when flying in an environment such as an urban canyon, where features remain in the fields of view for a very short period of time.

Thesis Supervisor: Prof. Emilio Frazzoli
Title: Associate Professor

# Acknowledgments

I would like to thank everyone who has supported me during my graduate studies at MIT as well as throughout my academic career. First and foremost I would like to thank my family for their endless love and support. Dad, Mom, and Robin, you have always been there to support me in everything I have undertaken. All of my accomplishments are a result of the opportunities you have given me. I Love You.

The exquisite undergraduate education I received at Harvey Mudd College has prepared me well for my career and has made me a skillful engineer. Thank you to the professors who gave me a strong engineering background. And thanks to the lifelong friends that continue to be supportive even though we are apart.

The support of Aurora Flight Sciences has been invaluable during my graduate career. Jim Paduano, Olivier Toupet, Josh Karch, and Donald Eng have been helpful along the way and have helped ensure that this project will continue. In addition to the many useful technical discussions, the financial support from Aurora during the first portion of this project was greatly appreciated. As was the support of the NDSEG/ASEE Fellowship during the second half of my time at MIT.

My research advisers Ray Sedwick and Emilio Frazzoli have been helpful with this project. And a number of students have aided me along the way - Josh Bialkowski, Jeong hwan Jeong, and Been Kim.

The ceaseless encouragement of the astronauts and supporters of the Astronaut Scholarship Foundation has been a great motivator. Knowing that the greatest American heroes believe in my success has been a constant motivator.

My friends in Boston have made the past two years a lot of fun and I thank them for their encouragement. My close group of MIT friends, those from Harvey Mudd who moved across the country with me, and all of my new friends who are always there to have fun. My roommates Anthony, Bobby, and Maximus who make it so much fun to go home. To the guys of AIMSense, you have been great. I learned a lot and had a lot of fun.

To all who have encouraged me - Thank You.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Accurate vehicle state estimation is crucial for controlling the motion of any vehicle. For an Unmanned Aerial Vehicle (UAV) to complete its mission it must have a reasonable estimate of its position, velocity, and attitude. A number of instruments are available for determining these parameters, each of which has its own strengths and weaknesses.

Standard aircraft navigation primarily utilizes measurements from a Global Positioning System (GPS) and an Inertial Measurement Unit (IMU) to estimate the vehicle's position and attitude. These systems combine in a complementary fashion with the strengths of one balancing the weaknesses of the other. A GPS system communicates with satellites to very accurately locate the position of a vehicle, but is useless for determining attitude and is not as useful for capturing high frequency motions. An IMU consists of a number of accelerometers that measure the translational acceleration of the vehicle and gyroscopes to measure the angular rates. An IMU is useful for capturing the fast dynamics of the vehicle and provides an invariably available measurement, however drift in the estimates can accumulate by integrating the acceleration errors over time. When GPS measurements are available, a GPS/IMU system is accurate in estimating the vehicle's motion. [20]

Another instrument that can be very useful for motion estimation is a camera. The field of computer vision related to motion estimation is known as egomotion. To date much of the research in this field has focused on estimation using vision only and

with little regard to computational requirements. For the application on a small UAV weight is at a premium, computational capabilities are limited, and an IMU is always available for fusion with the vision estimates. A navigation system that incorporates measurements from multiple vision sensors, an IMU, and GPS is robust to the failure of any one of those elements and allows the UAV to navigate in regions that may not be feasible with a standard GPS/IMU system. A vision system serves as a supplement to the low-drift position estimates available from GPS with the added advantage that it is also used to estimate attitude. Both of these systems are complementary to the capabilities of an IMU. Additionally, some of the shortcomings of vision sensors (primarily noise and scale-factor ambiguity) can be mitigated by the introduction of IMU measurements, while drift problems and lack of relative navigation information inherent to IMU measurements (specifically when GPS is denied or temporarily lost) can be overcome using vision. [20]

## 1.1 Previous Work

A great deal of work has been conducted in the field of estimating motion from structure, or visual odometry [16]. Closely related to the goal of estimating structure from motion, in which a known motion is used to map a structure, this field determines the position and orientation of a camera by analyzing the motion of fixed objects in the field of view. This is related to the field of Simultaneous Localization and Mapping (SLAM), in which an unknown environment is mapped and the vehicle is located within that map [5], [4]. The major difference between SLAM and motion from structure is that in SLAM one of the major goals is to produce an accurate mapping of the environment. Using vision-based motion estimation the structure of the world is immaterial, as long as it is either fixed or changing slowly.

There is an inherent difficulty that arises when attempting to calculate egomotion using a single camera. The issue of determining the range to an object is difficult to overcome, as well as a certain ambiguity that occurs between small translational motion and rotational motion. To help overcome the range difficulties it is often the

case that stereo vision is used. [4] Stereo cameras have the advantage that the distance to objects can be simply calculated. However, there is a limited range of usefulness for a stereo system. If an object is too far away it will appear in the same location on both images and no useful depth estimate can be calculated. This maximum range is related to the baseline distance between the two cameras. For many ground-based applications, stereo cameras prove to be adequate for the mission requirements. A small ground robot has a limited stereo baseline, and thus a limited range of valid stereo estimates. However, small ground vehicles usually travel at low speeds and in confined environments, meaning that the objects to be imaged are not incredibly far away. Alternatively, large ground vehicles that travel at a higher velocity may position the cameras further apart to increase the stereo range. These may not be feasible solutions for a small UAV.

Vision-based estimation has been used in a wide range of environments, including for visual odometry on the Mars rovers [11]. However, in most cases the environment has been fairly controlled, with controlled lighting and a constrained size. The lighting limitation is related to the difficulty identifying and tracking complex features. The size constraint is often related to the limitations imposed by stereo cameras. The ability to accurately conduct vision-based motion estimation in an outdoor, uncontrolled environment for an extended period of time is the ultimate objective of this field of research. Some notable work has been done using only a single camera, something that may prove more useful for UAVs than a stereo configuration. Soatto, et al have developed an algorithm to implement a single camera motion from structure technique [10]. Perhaps the most relevant work known to the author was done by Veth, at the Air Force Institute of Technology, and involves the fusion of a single camera with IMU measurements [21].

There has been some recent work examining the influence of a wider field of view on the quality of the egomotion estimate. To increase the quality of the motion estimate, Davison has suggested the use of a wide field of view camera. With such a camera, the features remain in the field of view for a longer period of time and are visible at more extreme angles. [3]. Another group has proposed using a compound eye, inspired by

the eyes of insects, to achieve the same effect [15]. By looking in multiple directions more useful features may be tracked in directions other than straight ahead.

## 1.2 Mission Description

The application of this system is a small UAV with limited payload and computational capability. The specific quantities that must be estimated for accurate navigation are vehicle position, velocity, and attitude. During the course of a mission it is assumed that accurate accelerations and rotation rates are available at all times from the IMU. These values are accurate, however small biases in the estimates translate into large long-term drift in the vehicle position and attitude estimates. Regardless of vehicle motion, these position and attitude estimates drift with time. In many applications GPS information is available for part of the mission, but may become lost, denied, or obscured for long periods of time, during which no position information is available. This GPS loss could be the result of poor weather, flying indoors or in an urban canyon, or a number of other causes. Vision information has the potential to provide relatively low-drift data from which to estimate vehicle egomotion during GPS outages. Many of the causes of lost or intermittent GPS signals are physical obstructions such as buildings or mountains. In most situations where an object is blocking the GPS signal there are features for the camera to track. Situations with poor feature density include flying over the ocean or a desert or at high altitudes - all situations where GPS is likely to be available.

The goal of this project is to develop a capability for a UAV to navigate in a GPS-denied environment for an extended period of time using vision. An extended Kalman filter is developed and tested that is capable of integrating measurements from up to three cameras oriented in a general position, an IMU, and a GPS when available.

## 1.3 Approach

This project can be separated cleanly into two parts. The first part involves the extraction of useful features from an image stream. Easily trackable features must be identified in an image and then accurately tracked into subsequent frames of a video. It is important that a feature in one image be accurately correlated with corresponding feature in a previous image. There is a wide body of work in this field of computer vision that is available in an open source library, *OpenCV*. [2] The basic feature identification and tracking algorithms are briefly discussed in Appendix B as well as a number of improvements to the basic algorithms that have been implemented. The features found in this first part of the project serve as the input to the second portion - state estimation. Some details of egomotion estimation are discussed in Chapter 2 and an extended Kalman filter is developed to perform egomotion estimation. Once the filter has been developed extensive testing and evaluation is performed in simulation, which is presented in Chapter 3.

## 1.4 Contribution

In addition to building a capability for vision-based navigation that fuses measurements from multiple cameras, an IMU, and GPS, the primary contribution of this thesis is a numerical investigation of various camera configuration choices and how they affect the quality of a motion estimate. This thesis will outline the development of such a capability as well as evaluate some of the scenarios in which using multiple cameras gains an advantage over a single camera. Some suggestions are made for the number and orientation of cameras in various situations.

## 1.5 Nomenclature

### Acronyms

| | |
|---|---|
| EKF | Extended Kalman Filter |
| FIM | Fisher Information Matrix |
| FOV | Field of View |
| GPS | Global Positioning System |
| IMU | Inertial Measurement Unit |
| SLAM | Simultaneous Localization and Mapping |
| $SO(3)$ | Special Orthogonal $3 \times 3$ matrix |
| UAV | Unmanned Aerial Vehicle |

### Variables

| | |
|---|---|
| $a_m^B$ | Measured acceleration in body axis |
| $b_a$ | Accelerometer bias |
| $b_g$ | Gyroscope bias |
| $f$ | Focal distance |
| $s_a$ | Accelerometer scale factor error |
| $s_g$ | Gyroscope scale factor error |
| $v$ | Translational velocity vector (meters per second) |
| $\mathbf{x}$ | Feature location on the image plane (2 dimensional) |
| $\hat{x}$ | State vector estimate |
| $y$ | Measurement from sensor (camera, IMU, GPS) |
| $\hat{y}$ | Measurement estimate |
| $\tilde{y}$ | Innovation or measurement residual |
| $\lambda$ | Depth to feature |
| $\sigma_{Measurement}$ | Covariance of vision measurement |
| $\sigma_\lambda$ | Covariance on depth estimate |
| $\sigma_v$ | Covariance on velocity |
| $\omega_m^B$ | Measured rotation rate in body axis |

| | |
|---|---|
| $F$ | State transformation matrix |
| $H$ | Measurement sensitivity matrix |
| $M$ | Initial covariance on feature depth |
| $N$ | Total number of features tracked by the filter |
| $N_A$ | Number of features tracked by the filter in camera A |
| $N_B$ | Number of features tracked by the filter in camera B |
| $N_C$ | Number of features tracked by the filter in camera C |
| $P$ | Covariance matrix |
| $R_{Cam}$ | Rotation matrix specifying camera orientation |
| $T$ | Translation vector from starting time to current time (meters) |
| $U$ | Initial covariance on IMU error states |
| $W$ | Initial covariance on velocity |
| $\mathbf{X}$ | Feature location in space (3 dimensional) |
| $\Sigma_w$ | Process error covariance |
| $\Sigma_n$ | Measurement error covariance |
| $\Omega$ | Rotation (attitude) vector from starting time to current time (radians) |

# Chapter 2

# State Estimation

This chapter provides a detailed description of the system implemented to estimate vehicle egomotion. The basic framework consists of an extended Kalman filter which combines measurements from a suite of cameras, an IMU, and GPS. The framework is an extension of the single-camera vision algorithm presented in *An Invitation To 3D Vision* [10]. A Kalman filter is well suited for combining measurements from a number of different instruments, each of which has an associated measurement uncertainty.

The state vector used for this implementation is given below. The states are chosen so that the filter tracks key location parameters of the features in the camera field of views, the principle vehicle states (position, velocity, attitude, and angular rates), and sensor error terms. The terms relating to feature location and sensor errors are used to refine the estimates of the principle vehicle states.

$$\mathbf{x} = \begin{pmatrix} \text{Initial Feature Location} \\ \text{Initial Feature Depth} \\ \text{Translation} \\ \text{Attitude} \\ \text{Velocity} \\ \text{Angular Rate} \\ \text{Accelerometer Errors} \\ \text{Gyroscope Errors} \end{pmatrix}$$

This state vector is represented by ten subvectors and is given by

$$\mathbf{x} = \begin{bmatrix} x_0 & \lambda & T^I & \Omega^I & v^I & \omega^I & s_a & b_a & s_g & b_g \end{bmatrix}^T \tag{2.1}$$

where $x_0$ is a vector containing the two dimensional coordinates of each feature on the image plane in the first frame, $\lambda$ is a vector containing the depth of each feature in the first frame, $T^I$ is the translation vector, $\Omega^I$ is the vector specifying the attitude, $v_I$ is the translational velocity vector, $\omega^I$ specifies the rotational velocity, $s_a$ is the accelerometer scale factor error, $b_a$ is the accelerometer bias, $s_g$ is the gyroscope scale factor error, and $b_g$ is the gyroscope bias

First, a basic description of egomotion estimation is given. Next, the various measurement models are described. Finally, a detailed description of the implemented extended Kalman filter is presented.

## 2.1 Rigid Body Motion

When observing the motion of the features in a camera view it is in general necessary to specify the trajectory of each feature. However, if it is assumed that each of the features is stationary in space and that the camera moves, then it is only necessary to specify the overall rigid body motion undergone by the feature space. Under rigid body motion the distance between any two features is constant. [14] Any rigid body motion can be represented by a single rotation and a single translation.

### 2.1.1 Assumption of Stationary Features

For this analysis it is assumed that all features in the world are stationary during the time they are visible. This is a necessity, as moving features do not provide information solely about the motion of the aircraft relative to the ground. In all simulations and experiments performed the world is stationary, however in a real world situation this may not be the case. Flying over a road, for instance, there will likely be moving cars or people in the field of view. This is a limitation of this

approach that may be corrected by carefully choosing features that appear to be stationary and by eliminating anomalous features from the filter state.

### 2.1.2 Geometry of Rotations

A rigid object rotating about a fixed point is shown in Figure 2-1. In general it may be assumed that the origin of the inertial frame is the center of rotation. If this is not the case the origin may be translated to the center of rotation. This rotation is

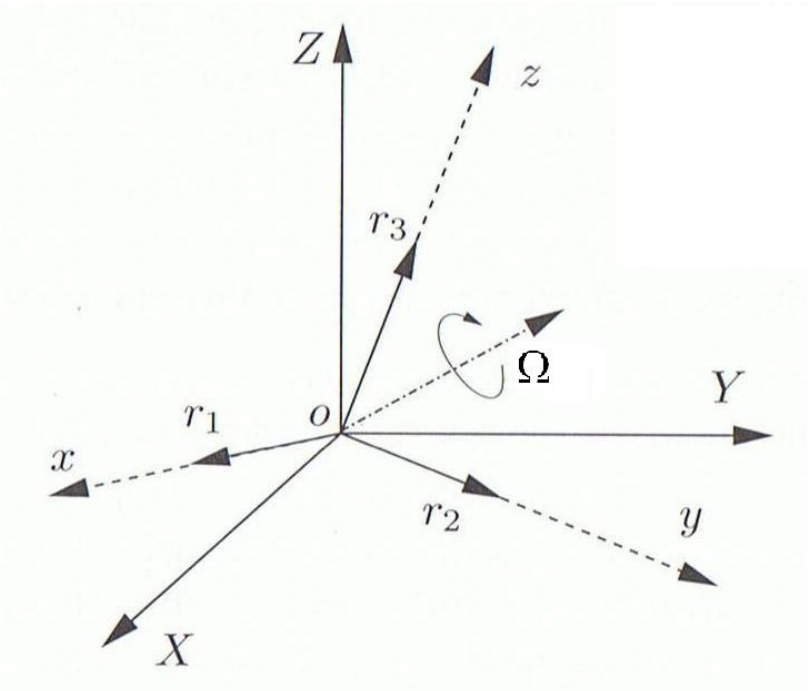


Figure 2-1: Rotation of a rigid body about a single axis [10]

represented by a rotation matrix $R$, which rotates the rigid body from the inertial coordinate frame XYZ to the rotating coordinate frame xyz. This rotation matrix is both orientation-preserving and orthogonal. Thus the rotation matrix is known as a *special orthogonal matrix*, where *special* indicates that the matrix is orientation preserving. The space of all such matrices is denoted by

$$SO\left(3\right) \doteq \left\{R \in \mathbb{R}^{3\times 3} | R^T R = I, \det\left(R\right) = +1\right\}. \tag{2.2}$$

Any rotation may be realized by rotating about some fixed axis by a certain angle. Let $\Omega_a \in \mathbb{R}^3$ be the unit vector specifying the axis of rotation and $\theta \in \mathbb{R}$ be the angle of rotation (in radians). For simplicity, let $\Omega_a$ be scaled by $\theta$ to obtain the single vector $\Omega \doteq \theta\Omega_a \in \mathbb{R}^3$ that contains all of the necessary information to build a rotation matrix. To obtain a relationship between the desired rotation matrix and the vector $\Omega$, let us consider the motion of a single point $q$ on a rotating body. [14] Rotating at a constant unit velocity about the axis $\Omega_a$, the velocity of this point is given by

$$\dot{q}(t) = \Omega_a \times q(t) = \Omega_a^{\times} q(t) \tag{2.3}$$

where $\Omega_a^{\times}$ is a skew symmetric matrix defined as

$$\Omega^{\times} \doteq \begin{bmatrix} 0 & -\Omega_3 & \Omega_2 \\ \Omega_3 & 0 & -\Omega_1 \\ -\Omega_2 & \Omega_1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times 3}. \tag{2.4}$$

$\Omega^{\times}$ is a member of the vector space of skew-symmetric matrices, often denoted as $so(3)$ (this space is the algebra of the group $SO(3)$):

$$so(3) \doteq \left\{ S \in \mathbb{R}^{3\times 3} | S^T = -S \right\}. \tag{2.5}$$

Integrating the linear time-invariant differential equation (2.3), the position of the point $q$ at some time $t > 0$ is

$$q(t) = e^{\Omega_a^{\times} t} q(0). \tag{2.6}$$

Now, if the rigid body is rotated at a unit velocity for $\theta$ seconds (or simply rotated by $\theta$ radians), the position of the point $q$ will be

$$q(\theta) = e^{\Omega_a^{\times} \theta} q(0) = e^{\Omega^{\times}} q(0). \tag{2.7}$$

The net rotation is given by

$$R = e^{\Omega^{\times}}. \tag{2.8}$$

Expanding the matrix exponential Taylor series and applying a few relations for powers of skew-symmetric matrices, a closed form solution for the rotation matrix is found. This solution, known as *Rodrigues' formula*, provides an efficient method for computing the rotation matrix specified by a given vector $\Omega$.

$$R = e^{\Omega^{\times}} = I + \frac{\Omega^{\times}}{||\Omega||}\sin\left(||\Omega||\right) + \left(\frac{\Omega^{\times}}{||\Omega||}\right)^{2}\left(1 - \cos\left(||\Omega||\right)\right). \tag{2.9}$$

Similarly, for any rotation matrix $R \in SO\left(3\right)$ there exists $\Omega \in \mathbb{R}^3$ such that $R = e^{\left(\Omega^{\times}\right)}$. However, such an $\Omega$ is not necessarily unique, e.g., as any vector of the form $\Omega' = \Omega + 2k\pi\frac{\Omega}{||\Omega||}$ with $k$ an integer would generate the same $R$ as $\Omega$. Since $R$ is formed by taking the matrix exponential of the skew symmetric matrix $\Omega^{\times}$, the inverse operation is referred to as the Logarithm on $SO\left(3\right)$, or the Inverse of Rodrigues' formula. If the rotation matrix is given by

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix},$$

then $\Omega$ is given by

$$\begin{aligned} ||\Omega|| &= \cos^{-1}\left(\frac{\text{trace}\left(R\right) - 1}{2}\right) \\ \frac{\Omega}{||\Omega||} &= \frac{1}{2\sin\left(||\Omega||\right)}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \end{aligned} \tag{2.10}$$

If $R = I$, then $||\Omega|| = 0$ and $\frac{\Omega}{||\Omega||}$ is undefined, and can be chosen arbitrarily [10].

## 2.2 Egomotion Estimation - Epipolar Geometry

Epipolar geometry refers to the study of views taken from cameras at distinct positions. These views could be from independent cameras at different positions or from a single camera at two instants in time after undergoing some motion. To be useful, most of the features from the first view must be visible in the second view. The relationship between the observed position of each of these features on the image planes places a constraint on the motion of the camera between frames, the *epipolar constraint*.

The case of a fixed point in space observed from two different perspectives is presented in Figure 2-2. The projections of point $p$ onto the image plane at the two different positions are given by $\mathbf{x}_1$ and $\mathbf{x}_2$. The rigid body transformation between the two camera views is given by $(R, T)$. Since $\mathbf{x}_1$ and $\mathbf{x}_2$ are projections of the same

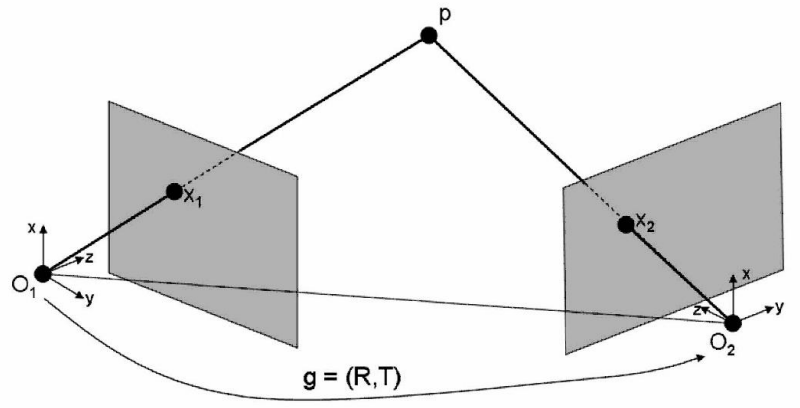


Figure 2-2: Feature projections onto rotated and translated image planes

point, transforming $\mathbf{x}_1$ into the reference frame of the second camera collocates the two projected points.

$$\mathbf{x}_2 T^{\times} R \mathbf{x}_1 = 0 \tag{2.11}$$

The matrix

$$E \doteq T^{\times} R \ \in \mathbb{R}^{3\times 3} \tag{2.12}$$

is the *essential matrix*, which encodes the relative pose between the two cameras [10].

**Recovering the Essential Matrix from Feature Measurements**

The essential matrix presented in Equation 2.12 may also be expressed as

$$E = \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix}, \tag{2.13}$$

or it may be expressed as the stacked matrix

$$E^s = \begin{bmatrix} e_{11} \\ e_{21} \\ e_{31} \\ e_{12} \\ e_{22} \\ e_{32} \\ e_{13} \\ e_{23} \\ e_{33} \end{bmatrix}. \tag{2.14}$$

The epipolar constraint may be rewritten as an inner product in the form

$$\mathbf{a}^T E^s = 0 \tag{2.15}$$

where $\mathbf{a} \doteq \mathbf{x}_1 \otimes \mathbf{x}_2$ is the Kronecker product of the two three dimensional vectors representing the feature locations at the two times. This is for a single feature only, to incorporate multiple feature measurements Equation 2.15 must be written as

$$\chi E^s = 0 \tag{2.16}$$

where

$$\chi \doteq \begin{bmatrix} \mathbf{a}^1 & \mathbf{a}^2 & \dots & \mathbf{a}^n \end{bmatrix}^T. \tag{2.17}$$

This linear system may be solved for the vector $E^s$. For the solution to be unique up to a scale factor there must be at least 8 features in a general position. In certain special cases, such as motion over a completely planar scene, this requirement may be relaxed to 4 features. [10], [13]

## 2.3 Scale Factor Ambiguity

When viewing an object through a projection camera, there is an inherent scale ambiguity. To illustrate this ambiguity imagine a square object that appears on the image plane as a $n$ by $n$ pixel region. This projection could be created by an object of size $m$ by $m$ at a distance $d$. Or, this same image could be created by an object of size $2m$ by $2m$ at a distance $2d$. Thus, without an estimate of the depth to an object, it is impossible to reconstruct its size.

A similar issue arises with translational motion in the field of view. An object translating parallel to the image plane at a distance of 10 meters and a velocity of 1 meter per second would have the same apparent velocity on the image plane as an object at a distance of 20 meters traveling at 2 meters per second. That is, objects further away appear to travel slower.

This phenomenon proves to be a major difficulty for egomotion estimation. If the camera observes features traveling with a certain pixel velocity, but there is no information about the distance to the feature, it is impossible to determine the linear velocity of the camera. It should be noted that the scale factor ambiguity is not a problem for cases of pure rotation. The projection of the features of different depths onto the optical plane is unaffected by angular displacements of the camera. If the distance to the features cannot be accurately determined, it is impossible to obtain a reliable egomotion estimate in translation. A number of different solutions may be used to obtain the depth to the features in the field of view. A few of those solutions are evaluated below.

### 2.3.1 Stereo Cameras

Binocular vision is widely used in nature to determine the distance to objects. Comparing images of an object taken from two different points of view allows for a crude estimate of the distance to that object. If the distance between the two eyes is known, as in Figure 2-3, and the difference in the bearing to the object is measured, the distance can be calculated. In this example, the distance to the object is given by $\frac{d}{\tan(\theta)}$.

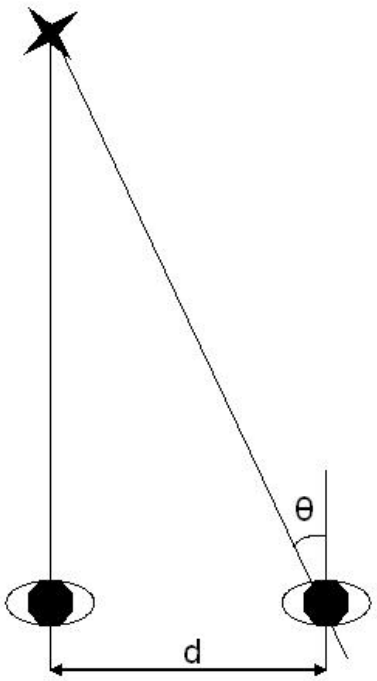


Figure 2-3: Stereo vision

A similar approach can be used with a pair of stereo cameras. The cameras are placed a known distance apart on the aircraft and are positioned such that the fields of view are largely overlapping. Sample views from stereo cameras are shown in Figure 2-4.

Note that the objects in Camera B are shifted slightly to the left of the same images in Camera A. Also note that not all of the objects have shifted by the same amount. Because of the same physical property described in Section 2.3, objects closer to the cameras such as the cat appear to move a greater distance in the optic plane

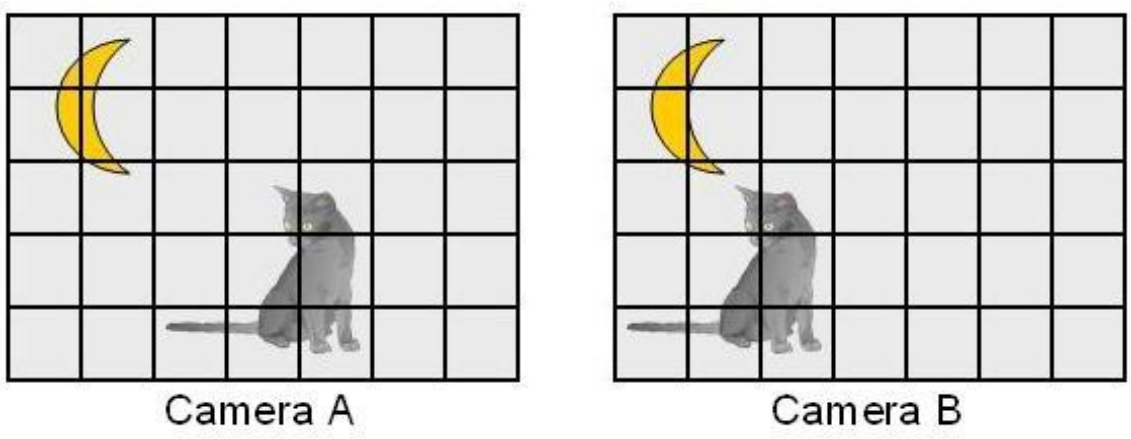


Figure 2-4: Stereo cameras

than the more distant moon. Knowing the baseline distance between the cameras $d$, the focal distance (the distance to the optical plane) $f$, and the number of pixels the feature is shifted by between cameras $p$, the distance to the object can be calculated.

$$Depth = \frac{fd}{p} \tag{2.18}$$

This can be applied to every feature in the field of view to provide reliable depth information for each feature.

### Advantages

Stereo cameras are very reliable for obtaining a decent depth estimate. Once the features have been located in each image, the calculations are fairly simple. For a large ground vehicle that is not limited by weight constraints, stereo vision is a very good solution to the scale ambiguity problem.

### Disadvantages

In a small, lightweight UAV stereo cameras may not be the best solution. This solution requires additional hardware in the form of an extra camera and the computing hardware necessary to process the information. If the UAV is payload limited, this extra weight budget may not be available. Existing image processing algorithms that

identify features and perform the correspondence problem to track features between frames are fairly computationally intensive. Adding a second camera requires identifying features in the second camera as well as performing the correspondence problem to match the features in the two camera views. This is essentially doubling the computational overhead of the process. The small size of a UAV limits the allowable distance between the cameras. If the cameras are close together, the depth estimate is not as reliable for distant features. Take the limiting case where the cameras are separated by zero distance. In this situation, the features in both images will be completely overlapping. If the cameras are then placed only one centimeter apart it should be obvious that the features very close to the camera will appear separated by a few pixels, but the features a few meters away will be positioned almost identically. Thus this small allowable baseline limits the potential range to the features. If the observed distance between features falls below the noise threshold (about one pixel), the stereo camera solution is not feasible. Additionally, the cameras must be carefully placed on the UAV and calibrated so that the exact baseline distance between the cameras is known. A long, rigid baseline is difficult to obtain on a small aircraft.

### 2.3.2 Kalman Estimator

An alternative solution for solving the scale factor ambiguity problem is to use an extended Kalman filter to estimate the depth to each feature. A review of the basic framework of an extended Kalman filter is presented in Appendix A. The details of the filter are described in Section 2.5, but the basic approach to finding the depth is to compare the estimated position of each feature to the observed position and then update the depth estimate accordingly. Since the system dynamics are nonlinear, the depths must be initialized fairly close to the truth. This initialization is accomplished with the help of GPS for the first few seconds to estimate the depth of each feature.

### Advantages

This solution is well suited to the problems posed by a small UAV. No additional hardware is necessary in the form of additional cameras or range finding equipment. Only a marginal increase in computing power may be necessary to accommodate a larger state vector. Additionally, there is no inherent limitation imposed by physical vehicle size.

### Disadvantages

The major limitation of this approach is that the system dynamics are nonlinear and if the depths are initialized too far from the true values they may not converge to reality. Accurate initialization relies on precise knowledge of vehicle position, the very quantity that is being estimated by the filter. An assumption that is made for this application is that the filter was started during a period where GPS was available. This could be before the UAV drops below rooftop level or enters a building. If GPS is available when the feature locations are initialized, and then if features enter and leave the field of view in a continuous manner, the initialized depth should be sufficient to ensure that the filter converges.

## 2.4 Sensors

A sensor model is necessary to accurately create the measurement equation to update the Kalman filter. Each type of sensor will have a different measurement equation. This section presents the models of each type of sensor used in this filter, as well as the derivations of their measurement equations. The formulation of the IMU is based in part on the MIT Masters thesis of Stephen Paschall [7], and the single camera vision filter is based on the filter presented in *An Invitation to 3D Vision* [10].

### 2.4.1 Inertial Measurement Unit

#### Accelerometer Modeling

This model uses strapdown accelerometers, which means that they measure accelerations in the body reference frame. The measured acceleration in the body frame is given by:

$$a_m^B = (I + \Gamma_a)(I + S_a)(a^B + b_a + \epsilon_a), \tag{2.19}$$

where $\Gamma_a$ represents misalignment errors, $S_a$ the scale factor error, $b_a$ the accelerometer bias, and $\epsilon_a$ the accelerometer white noise with covariance $Q_a$. $a^B$ represents the true vehicle acceleration. For the purposes of this model, misalignment errors are neglected ($\Gamma_a = 0$). Neglecting second order error terms, the accelerometer error model becomes:

$$a_m^B \approx (I + S_a)a^B + b_a + \epsilon_a. \tag{2.20}$$

This model holds for each of the three orthogonal accelerometers. Representing the acceleration in each of the directions in matrix form with dimension three we get

$$\mathbf{a_m^B} \approx (I + S_a)\mathbf{a^B} + \mathbf{b_a} + \epsilon_\mathbf{a}, \tag{2.21}$$

where

$$\mathbf{a_m^B} = \begin{pmatrix} a_{m_x}^B \\ a_{m_y}^B \\ a_{m_z}^B \end{pmatrix} S_a = \begin{bmatrix} S_{a_x} & 0 & 0 \\ 0 & S_{a_y} & 0 \\ 0 & 0 & S_{a_z} \end{bmatrix} \mathbf{a^B} = \begin{pmatrix} a_x^B \\ a_y^B \\ a_z^B \end{pmatrix} \mathbf{b_a} = \begin{pmatrix} b_{a_x} \\ b_{a_y} \\ b_{a_z} \end{pmatrix} \epsilon_\mathbf{a} = \begin{pmatrix} \epsilon_{a_x} \\ \epsilon_{a_y} \\ \epsilon_{a_z} \end{pmatrix}$$

**Formulation for a Kalman filter**

To be useful in a Kalman filter, Equation 2.20 must be expressed as the true acceleration in terms of the measured acceleration and a linear combination of the estimated values. Rearranging,

$$\mathbf{a^B} \approx (I + S_a)^{-1}(\mathbf{a_m^B} - \mathbf{b_a} - \epsilon_\mathbf{a}) \tag{2.22}$$

For small scale factor errors,

$$(I + S_a)^{-1} \approx I - S_a \tag{2.23}$$

So,

$$\mathbf{a^B} \approx (I - S_a)(\mathbf{a_m^B} - \mathbf{b_a} - \epsilon_\mathbf{a}) = \mathbf{a_m^B} - \mathbf{b_a} - \epsilon_\mathbf{a} - S_a\mathbf{a_m^B} + S_a\mathbf{b_a} + S_a\epsilon_\mathbf{a} \tag{2.24}$$

Again, neglecting second order error terms,

$$\mathbf{a^B} \approx \mathbf{a_m^B} - \mathbf{b_a} - \epsilon_\mathbf{a} - S_a\mathbf{a_m^B}. \tag{2.25}$$

And since in the filter all of the estimated states must be vectors, we rewrite $S_a\mathbf{a_m}$ as $D_a\mathbf{s_a}$, where

$$D_a = \begin{bmatrix} a_{m_x} & 0 & 0 \\ 0 & a_{m_y} & 0 \\ 0 & 0 & a_{m_z} \end{bmatrix} \mathbf{s_a} = \begin{pmatrix} s_{a_x} \\ s_{a_y} \\ s_{a_z} \end{pmatrix}$$

To obtain the best estimate of the true acceleration, we take the expected value of Equation 2.25.

$$\hat{\mathbf{a}}^\mathbf{B} = \mathbf{a_m^B} - \hat{\mathbf{b}}_\mathbf{a} - D_a\hat{\mathbf{s}}_\mathbf{a} \tag{2.26}$$

Measurements from the IMU are all given in the body reference frame. However, the elements of the state vector presented in Equation 2.1 are given in the inertial coordinate system, so a transformation must be performed to change reference frames. $\Omega$ represents the rotation from the inertial coordinates to the body reference frame. Applying the rotation $R_{I\to B} = e^{\hat{\Omega}^{\times}(t)}$ rotates to the correct attitude. The IMU measurements can be transformed into the inertial reference frame by applying the inverse of this transform $R_{B\to I} = \left(e^{\hat{\Omega}^{\times}(t)}\right)^{-1}$. Note that since rotation matrices are orthogonal, the inverse is equal to the transpose.

$$R_{B\to I} = (R_{I\to B})^{-1} = R_{I\to B}^{T} \tag{2.27}$$

Taking the transpose of a matrix is considerably faster than taking the inverse. Thus, whenever it is necessary to take the inverse of a rotation matrix it is prudent to take the transpose instead. So, the measurements of acceleration derived above are transformed into the inertial reference frame by

$$\hat{\mathbf{a}}^{\mathbf{I}} = \left(e^{\hat{\Omega}^{\times}}\right)^{T} \hat{\mathbf{a}}^{\mathbf{B}} \left(\hat{\mathbf{s}}_{\mathbf{a}}, \hat{\mathbf{b}}_{\mathbf{a}}, \mathbf{a}_{\mathbf{m}}^{\mathbf{B}}\right) \tag{2.28}$$

### Gyroscope Modeling

The formulation for gyroscope model parallels that of the accelerometer. The gyroscope error model is given by

$$\omega^{\mathbf{B}} = \omega_{\mathbf{m}}^{\mathbf{B}} - \mathbf{b}_{\mathbf{g}} - \epsilon_{\mathbf{g}} - D_g \mathbf{s}_{\mathbf{g}}. \tag{2.29}$$

where

$$\omega^{\mathbf{B}} = \begin{pmatrix} \omega_x^B \\ \omega_y^B \\ \omega_z^B \end{pmatrix} \omega_{\mathbf{m}}^{\mathbf{B}} = \begin{pmatrix} \omega_{m_x}^B \\ \omega_{m_y}^B \\ \omega_{m_z}^B \end{pmatrix} \mathbf{b}_{\mathbf{g}} = \begin{pmatrix} b_{g_x} \\ b_{g_y} \\ b_{g_z} \end{pmatrix} \epsilon_{\mathbf{g}} = \begin{pmatrix} \epsilon_{g_x} \\ \epsilon_{g_y} \\ \epsilon_{g_z} \end{pmatrix}$$

$$D_g = \begin{bmatrix} \omega_{m_x} & 0 & 0 \\ 0 & \omega_{m_y} & 0 \\ 0 & 0 & \omega_{m_z} \end{bmatrix} \mathbf{s}_{\mathbf{g}} = \begin{pmatrix} s_{g_x} \\ s_{g_y} \\ s_{g_z} \end{pmatrix}$$

To obtain the best estimate of the true rotation rates, we take the expected value of Equation 2.29.

$$\hat{\omega}^{\mathbf{B}} = \omega_{\mathbf{m}}^{\mathbf{B}} - \hat{\mathbf{b}}_{\mathbf{g}} - D_g \hat{\mathbf{s}}_{\mathbf{g}} \tag{2.30}$$

Just as is done with the accelerations, the rotation rates are transformed into inertial coordinates by

$$\hat{\omega}^{\mathbf{I}} = \left(e^{\hat{\Omega}^{\times}}\right)^T \hat{\omega}^{\mathbf{B}} \left(\hat{\mathbf{s}}_{\mathbf{g}}, \hat{\mathbf{b}}_{\mathbf{g}}, \omega_{\mathbf{m}}^{\mathbf{B}}\right) \tag{2.31}$$

The egomotion estimates ($T(t)$ and $\Omega(t)$) obtained are the displacement of the camera between time zero and the current time. This means that at time zero the camera reference frame is aligned with the egomotion estimate coordinates.

### Measurement Equations For Filter Update

The state vector and covariance are updated as described in Section A.0.3 with a measurement from the IMU, the measurement equation $h_{\text{IMU}}(.)$ and the measurement sensitivity matrix $H_{\text{IMU}}(t)$. The measurement from the IMU is a concatenation of the measured acceleration and the measured rotation rates.

$$y(t+1) = \begin{bmatrix} \mathbf{a}_{\mathbf{m}}^{\mathbf{B}} \\ \omega_{\mathbf{m}}^{\mathbf{B}} \end{bmatrix} \tag{2.32}$$

and the measurement equation is the estimated value of the measurement

$$\hat{y} = h_{\text{IMU}}(\hat{x}(t+1|t)) = \begin{bmatrix} \hat{a}_m^B \\ \hat{\omega}_m^B \end{bmatrix} = \begin{bmatrix} a_m^B - D_a \hat{s}_a - \hat{b}_a \\ \omega_m^B - D_g \hat{s}_g - \hat{b}_g \end{bmatrix} \tag{2.33}$$

The measurement sensitivity matrix is the linearization of the measurement equation.

$$H_{\text{IMU}}(t+1) = \frac{\partial \hat{y}}{\partial \hat{x}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & -D_a & -I & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -D_g & -I \end{bmatrix} \tag{2.34}$$

### 2.4.2 Global Positioning System

The capability to incorporate GPS updates is built into the Kalman filter for a number of reasons. When operating in the presence of GPS it is reasonable to incorporate the available measurements. Additionally, the GPS measurements may be used to initialize the vision system. A known trajectory is extremely useful for determining the distance to the features in the field of view, and this known trajectory can be obtained from a GPS fix. It is assumed that in practice the vision system will be initialized and the initial feature depths determined while in the presence of GPS or while following some known trajectory.

#### Measurement Equation For Filter Update

The measurement from a GPS unit consists of the three dimensional position and linear velocity of the vehicle. This measurement is obtained from the GPS unit even in the absence of a solid GPS signal. A covariance is provided along with the estimate and in the absence of a good position estimate the covariance will be very large. A large covariance indicates a lack of confidence in the estimated value, and the main Kalman filter will largely ignore this new measurement. Formally, the measurement from the GPS is given by

$$y\left(t+1\right) = \begin{bmatrix} \mathbf{T} \\ \\ \mathbf{v} \end{bmatrix} \tag{2.35}$$

and the measurement equation is the estimated value of the measurement

$$\hat{y} = h_{\text{GPS}}\left(\hat{x}\left(t+1|t\right)\right) = \begin{bmatrix} \hat{\mathbf{T}} \\ \\ \hat{\mathbf{v}} \end{bmatrix} \tag{2.36}$$

The measurement sensitivity matrix is the linearization of the measurement equation and is given by

$$H_{\text{GPS}}\left(t+1\right) = \frac{\partial \hat{y}}{\partial \hat{x}} = \begin{bmatrix} 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.37}$$

### 2.4.3 Vision Sensor

A vision sensor is a camera rigidly mounted to the vehicle structure that is capable of determining the two dimensional location of features in its field of view. It is assumed that some processing has already been performed to reduce the entire camera image to a relatively small number of generic features. Additionally, the features must be individually tagged with a unique feature identifier that is consistent as long as the feature is in the camera view.

The camera reference frame is right handed with the origin located at the center of an image. The x-axis is to the right, the y-axis is down, and the z-axis is forward (in the direction the camera is pointing). Objects in the field of view are observed by the camera as projections onto the image plane.

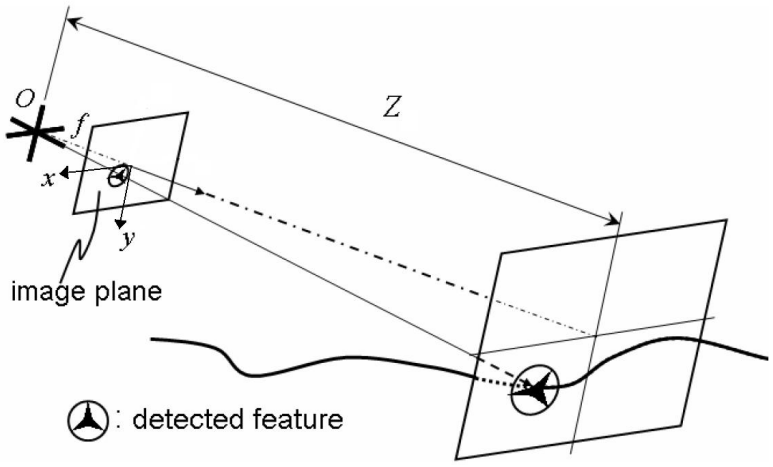


Figure 2-5: Camera reference frame [20]

The camera is observing features in a three dimensional world that in general are not all at the same distance from the camera. A single camera is not capable of distinguishing the three dimensional position of the features, it can only measure the two dimensional position. The three dimensional position of a generic feature $i$ relative to the camera reference frame is given by the vector $X^i$ drawn between the

center of the camera and the feature.

$$X^i = \begin{bmatrix} X_1^i \\ X_2^i \\ X_3^i \end{bmatrix} \tag{2.38}$$

Although there may be multiple cameras oriented in different directions, with different reference frames, the feature locations are not converted to any consistent coordinate system. The features of each camera are maintained in the reference frame of that camera. The two dimensional representation of this feature is the projection onto the image plane of a camera. For a given camera, the image plane is located at the focal length. Reasonable focal lengths for cameras that may be used in this application are around 6 millimeters. In this framework the image plane is located one meter from the focal point. The $\pi$ operator is the projection of the three dimensional feature position onto the image plane and is defined by

$$\mathbf{x}^i = \begin{bmatrix} x_1^i \\ x_2^i \end{bmatrix} = \pi\left(\mathbf{X}^i\right) \doteq \begin{bmatrix} \frac{X_1^i}{X_3^i} \\ \frac{X_2^i}{X_3^i} \end{bmatrix} \tag{2.39}$$

These coordinates can also be expressed with a third element that is equal to one. This represents the features on the image plane in the same coordinates used to define the points in 3D space.

$$\mathbf{x}^i = \begin{bmatrix} x_1^i \\ x_2^i \\ 1 \end{bmatrix} \tag{2.40}$$

The choice to work with a focal length of one meter is made for convenience. The depths estimated by the filter are in units of multiples of the focal length. An object 10 meters away would have an estimated depth of 10 using a one meter focal length and an estimated depth of 20 using a 50 centimeter focal length. The choice of focal length does not effect the outcome of the motion estimation, as the image plane coordinates are scaled by the same amount. It can be demonstrated that as

long as the angular field of view is held constant, the same image is generated by cameras of varying focal length. With this result it is simple to convert any camera measurements to data on the canonical unit image plane. Measurements from the camera come as pixels, or fractions of the image plane. Even without knowledge of the true focal length, these values can easily be scaled to the appropriate size image plane at a distance of one meter.

### Vision Sensor Modeling

Each of the cameras on the vehicle is pointed in a different direction, the orientation of which is specified by a rotation matrix. The rotation matrix represents the rotation undergone to rotate the camera from a downward looking camera (with the top of the image plane toward the front of the aircraft). For instance, a forward looking camera would be specified as a rotation by $+90^\circ$ about the camera's x-axis. The vectors corresponding to the rotation matrix for each cardinal camera direction are given in Table 2.1. A downward looking camera was chosen as the basis direction because it is the direction most likely to be used.

Table 2.1: Orientation rotations for camera directions

| Camera Direction | Rotation vector |
|---|---|
| Forward | $\left[\frac{\pi}{2},\ 0,\ 0\right]$ |
| Right | $\left[\frac{2\pi}{3\sqrt{3}},\ \frac{2\pi}{3\sqrt{3}},\ \frac{2\pi}{3\sqrt{3}}\right]$ |
| Left | $\left[\frac{2\pi}{3\sqrt{3}},\ -\frac{2\pi}{3\sqrt{3}},\ -\frac{2\pi}{3\sqrt{3}}\right]$ |
| Down | $[0,\ 0,\ 0]$ |
| Up | $[\pi,\ 0,\ 0]$ |
| Back | $\left[0,\ \frac{\pi}{\sqrt{2}},\ \frac{\pi}{\sqrt{2}}\right]$ |

This model assumes that each camera is rigidly connected to the aircraft frame and undergoes a rigid body motion consisting of a single rotation and a single translation. For ease of understanding, it is initially assumed that each camera is stationary and the world rotates and translates around the camera. Each feature is represented by three Cartesian coordinates in the camera's reference frame, represented by $\mathbf{X}^i(t)$.

At any time, the three dimensional location of a feature is related to the location at time zero as well as the rotation and translation undergone during that time period.

$$\mathbf{X}^i(t) = e^{\mathbf{\Omega}_\mathbf{F}^\times(t)} x_0^i(t)\,\lambda^i(t) + \mathbf{T}_F(t) \tag{2.41}$$

Note that in the above equation the rotation and translation vectors are denoted with the subscript 'F' denoting that this is the motion of the features around the stationary camera. These motion vectors, as well as the feature coordinates, are in the reference frame of the camera. The values estimated in the state vector, which are represented by $\mathbf{\Omega}$ and $\mathbf{T}$ with no subscript, refer to the motion of the aircraft/camera and assume that the features are stationary in the world. The relationship between these two notations is a simple negative sign and a rotation.

$$\begin{aligned} \mathbf{T}_F &= -R_{\text{Cam}}^T \mathbf{T} \\ \mathbf{\Omega}_F &= -R_{\text{Cam}}^T \mathbf{\Omega} \end{aligned} \tag{2.42}$$

If a single camera extended Kalman filter is implemented with no other measurement sources (IMU, GPS, etc), as in [10], the distinction of feature motion as compared to camera motion is not crucial. Since a simple sign inversion can be used to determine the vehicle motion from the feature motion it makes no difference which is estimated. When additional measurements are incorporated into the filter it is important that the same values are estimated by each sensor. GPS and IMU sensors naturally provide updates of the vehicle motion, so it makes sense for the vision sensor to use the same convention. In all future discussion and implementations Equation 2.41 shall be represented as

$$\mathbf{X}^i(t) = e^{-\mathbf{R}_{\text{Cam}}^\mathbf{T}\mathbf{\Omega}^\times(t)} x_0^i(t)\,\lambda^i(t) - R_{\text{Cam}}^T \mathbf{T}(t) \tag{2.43}$$

**Measurement Equations For Filter Update**

The state vector and covariance are updated as described in Section A.0.3 with a measurement from the camera, the measurement equation $h_{\text{Vision}}(.)$ and the measurement sensitivity matrix $H_{\text{Vision}}(t)$. The measurement from the vision sensor is a concatenation of the measured two dimensional positions of each feature in the field of view of the cameras.

$$\mathbf{y}(t+1) = \begin{bmatrix} \mathbf{x}_A^1(t+1) \\ \mathbf{x}_A^2(t+1) \\ . \\ . \\ \mathbf{x}_A^{NA}(t+1) \\ \mathbf{x}_B^1(t+1) \\ \mathbf{x}_B^2(t+1) \\ . \\ . \\ \mathbf{x}_B^{NB}(t+1) \\ \mathbf{x}_C^1(t+1) \\ \mathbf{x}_C^2(t+1) \\ . \\ . \\ \mathbf{x}_C^{NC}(t+1) \end{bmatrix} \tag{2.44}$$

The latest measurements from each camera are concatenated together, where the subscripts $A$, $B$, and $C$ indicate which camera the features are visible in. If fewer than three cameras are being used, those elements would be empty. Note that each element $\mathbf{x}^i(t+1)$ is a two dimensional column vector that contains the $x$ and $y$ coordinates of the feature. From Equation 2.43 it is known what the three dimensional location of each feature in the camera field of view is. The actual measurement is the projection of this feature location onto the image plane. Thus, the measurement equation is

given by

$$\hat{\mathbf{y}} = h\left(\hat{\mathbf{x}}\left(t+1|t\right)\right) = \begin{bmatrix} \pi\left(e^{-R_{\mathrm{CamA}}^{T}\boldsymbol{\Omega}^{\times}(t)}\mathbf{x}_{0A}^{1}\left(t\right)\lambda_{A}^{1}\left(t\right) - R_{\mathrm{CamA}}^{T}\mathbf{T}\left(t\right)\right) \\ \pi\left(e^{-R_{\mathrm{CamA}}^{T}\boldsymbol{\Omega}^{\times}(t)}\mathbf{x}_{0A}^{2}\left(t\right)\lambda_{A}^{2}\left(t\right) - R_{\mathrm{CamA}}^{T}\mathbf{T}\left(t\right)\right) \\ . \\ . \\ \pi\left(e^{-R_{\mathrm{CamA}}^{T}\boldsymbol{\Omega}^{\times}(t)}\mathbf{x}_{0A}^{NA}\left(t\right)\lambda_{A}^{NA}\left(t\right) - R_{\mathrm{CamA}}^{T}\mathbf{T}\left(t\right)\right) \\ \pi\left(e^{-R_{\mathrm{CamB}}^{T}\boldsymbol{\Omega}^{\times}(t)}\mathbf{x}_{0B}^{1}\left(t\right)\lambda_{B}^{1}\left(t\right) - R_{\mathrm{CamB}}^{T}\mathbf{T}\left(t\right)\right) \\ \pi\left(e^{-R_{\mathrm{CamB}}^{T}\boldsymbol{\Omega}^{\times}(t)}\mathbf{x}_{0B}^{2}\left(t\right)\lambda_{B}^{2}\left(t\right) - R_{\mathrm{CamB}}^{T}\mathbf{T}\left(t\right)\right) \\ . \\ . \\ \pi\left(e^{-R_{\mathrm{CamB}}^{T}\boldsymbol{\Omega}^{\times}(t)}\mathbf{x}_{0B}^{NB}\left(t\right)\lambda_{B}^{NB}\left(t\right) - R_{\mathrm{CamB}}^{T}\mathbf{T}\left(t\right)\right) \\ \pi\left(e^{-R_{\mathrm{CamC}}^{T}\boldsymbol{\Omega}^{\times}(t)}\mathbf{x}_{0C}^{1}\left(t\right)\lambda_{C}^{1}\left(t\right) - R_{\mathrm{CamC}}^{T}\mathbf{T}\left(t\right)\right) \\ \pi\left(e^{-R_{\mathrm{CamC}}^{T}\boldsymbol{\Omega}^{\times}(t)}\mathbf{x}_{0C}^{2}\left(t\right)\lambda_{C}^{2}\left(t\right) - R_{\mathrm{CamC}}^{T}\mathbf{T}\left(t\right)\right) \\ . \\ . \\ \pi\left(e^{-R_{\mathrm{CamC}}^{T}\boldsymbol{\Omega}^{\times}(t)}\mathbf{x}_{0C}^{NC}\left(t\right)\lambda_{C}^{NC}\left(t\right) - R_{\mathrm{CamC}}^{T}\mathbf{T}\left(t\right)\right) \end{bmatrix}. \tag{2.45}$$

The measurement sensitivity matrix is the linearization of the measurement equation,

$$H\left(t+1\right) = \frac{\partial\hat{\mathbf{y}}}{\partial\hat{x}} = \begin{bmatrix} H_{A}^{1} \\ H_{A}^{2} \\ . \\ . \\ H_{j}^{i} \\ . \\ . \\ H_{C}^{NC} \end{bmatrix} \tag{2.46}$$

where $j = A, B, C$ refers to the camera the feature is found in and $i = 1, 2, ..., Nj$ refers to the identifying tag. The submatrices in the measurement sensitivity matrix

are given by

$$H_j^i = \frac{\partial \mathbf{x}^i}{\partial \mathbf{X}_j^i} \frac{\partial X_j^i}{\partial x} \doteq \Pi_j^i \frac{\partial \mathbf{X}^i}{\partial x} \tag{2.47}$$

where

$$\mathbf{X}_j^i(t) \doteq e^{-R_{\mathrm{Cam}j}^T \Omega^{\times}(t)} x_{0j}^i(t) \lambda_j^i(t) - R_{\mathrm{Cam}j}^T T(t) \tag{2.48}$$

$$\Pi_j^i = \frac{1}{Z_j^i} \begin{bmatrix} I_2 & -\pi\left(\mathbf{X}_j^i\right) \end{bmatrix} \tag{2.49}$$

$$Z_j^i(t) \doteq \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \mathbf{X}_j^i(t) \tag{2.50}$$

$$\frac{\partial \mathbf{X}_j^i}{\partial x} = \left[ \underbrace{\left[0, ..., \frac{\partial \mathbf{X}_j^i}{\partial x_0^i}, ..., 0\right]}_{3 \text{ x } (2N-6m)}, \underbrace{\left[0, ..., \frac{\partial \mathbf{X}_j^i}{\partial \lambda^i}, ..., 0\right]}_{3 \text{ x } (N-m)}, \underbrace{\frac{\partial \mathbf{X}_j^i}{\partial T}}_{3 \text{ x } 3}, \underbrace{\frac{\partial \mathbf{X}_j^i}{\partial \Omega}}_{3 \text{ x } 3}, \underbrace{0}_{3 \text{ x } 18} \right] \tag{2.51}$$

where $m$ is the number of cameras and $N$ is the total number of features tracked in all cameras. The partial derivatives are given by

$$\frac{\partial \mathbf{X}_j^i}{\partial x_0^i} = e^{-R_{\mathrm{Cam}j}^T \Omega^{\times}} \begin{bmatrix} I_2 \\ 0 \end{bmatrix} \lambda_j^i \tag{2.52}$$

$$\frac{\partial \mathbf{X}_j^i}{\partial \lambda_j^i} = e^{-R_{\mathrm{Cam}j}^T \Omega^{\times}} x_0^i \tag{2.53}$$

$$\frac{\partial \mathbf{X}_j^i}{\partial T} = -R_{\mathrm{Cam}j}^T \tag{2.54}$$

$$\frac{\partial \mathbf{X}_j^i}{\partial \Omega} = -\begin{bmatrix} \frac{\partial e^{-R_{\mathrm{Cam}j}^T \Omega^{\times}}}{\partial \Omega_1} x_0^i \lambda_j^i & \frac{\partial e^{-R_{\mathrm{Cam}j}^T \Omega^{\times}}}{\partial \Omega_2} x_0^i \lambda_j^i & \frac{\partial e^{-R_{\mathrm{Cam}j}^T \Omega^{\times}}}{\partial \Omega_3} x_0^i \lambda_j^i \end{bmatrix} \tag{2.55}$$

## 2.5 Vision, IMU, & GPS Extended Kalman Filter

The general framework for an extended Kalman filter described in Appendix A is now presented for the specific filter implemented. As a reminder, the state vector is repeated below.

$$\mathbf{x} = \begin{bmatrix} x_0 & \lambda & T^I & \Omega^I & v^I & \omega^I & s_a & b_a & s_g & b_g \end{bmatrix}^T \tag{2.1}$$

Feature coordinates from all cameras are included in the $x_0$ and $\lambda$ terms. In general, the features are listed in the state vector in the order they are introduced to the filter, without regard to which camera they are measured in. Each feature has a unique identification tag as well as a tag indicating which camera it is in. It is assumed that the cameras have non-overlapping fields of view such that each feature is visible in only one camera. If the fields of view overlap and a feature is visible in both images, the correlation of these features is not made. The features are taken to be two independent objects and no additional information is assumed.

In general, the various sensors being used will provide measurements at different rates. This means that to properly incorporate all available information, the filter must run at a rate at least as fast as the fastest sensor. Between measurements the filter state and covariance are propagated according to the current state information and the model of state dynamics, as described in section A.0.2. When a measurement from a sensor becomes available, the state and covariance are updated according to the new information. Multiple measurements may be incorporated at the same time by simply applying the updates sequentially.

### 2.5.1 State Propagation

Although the measurement equations for each sensor described in section 2.4 are different for each type of sensor, the prediction step of the Kalman filter is largely the same regardless of the sensor type. The prediction step uses the current state estimate to propagate the state vector and state error covariance forward in time.

$$\hat{\mathbf{x}}\left(t+1|t\right) = f\left(\hat{\mathbf{x}}\left(t|t\right)\right) \tag{2.56}$$

In the absence of an IMU, the propagation step is given by

$$\begin{aligned}
x_0^i\left(t+1|t\right) &= x_0^i\left(t|t\right) \qquad i = 4, 5, ..., N_A + N_B + N_C \\
\lambda^i\left(t+1|t\right) &= \lambda^i\left(t|t\right) \qquad i = 2, 3, ..., N_A + N_B + N_C \\
T\left(t+1|t\right) &= e^{\omega^{\times}(t|t)} T\left(t|t\right) + v\left(t|t\right) \\
\Omega\left(t+1|t\right) &= \log_{SO(3)}\left[e^{\omega^{\times}(t|t)} e^{\Omega^{\times}(t|t)}\right] \\
v\left(t+1|t\right) &= v\left(t|t\right) \\
\omega\left(t+1|t\right) &= \omega\left(t|t\right) \\
s_a\left(t+1|t\right) &= s_a\left(t|t\right) \\
b_a\left(t+1|t\right) &= b_a\left(t|t\right) \\
s_g\left(t+1|t\right) &= s_g\left(t|t\right) \\
b_g\left(t+1|t\right) &= b_g\left(t|t\right)
\end{aligned} \tag{2.57}$$

Note that the only terms that are changed in this propagation step are the terms referring to position and attitude. The estimates of the initial feature location at time zero are not changed because an accurate estimate of these values can only be obtained through a measurement update. The linear velocity $v$ and angular velocity $\omega$ are assumed to be constant. The assumption is that changes in the vehicle trajectory begin as white noise disturbances to the velocities. Finally, the IMU error terms are assumed to be constant.

The only difference for an IMU update is in the propagation of the linear velocity $v$ and the angular rates $\omega$. The vision model assumes a white noise disturbance on the velocity, whereas the IMU model assumes a white noise disturbance on the accelerations. Acceleration is not included in the state vector, yet the most recent acceleration measurement is used in the propagation step to replace the terms presented above.

$$\begin{aligned} v\left(t+1|t\right) &= v\left(t|t\right) + \left(e^{\Omega^{\times}(t)}\right)^{T} \hat{a}^{B}\left(\hat{s}_{a}\left(t\right), \hat{b}_{a}\left(t\right), a_{m}^{B}\left(t\right)\right) \\ \omega\left(t+1|t\right) &= \left(e^{\Omega^{\times}(t)}\right)^{T} \omega^{B\times}\left(\hat{s}_{g}\left(t\right), \hat{b}_{g}\left(t\right), \omega_{m}^{B}\left(t\right)\right) \end{aligned} \tag{2.58}$$

### 2.5.2 Covariance Propagation

The state transformation matrix $F$ is found by linearizing the state function (Equation 2.56). In the absence of an IMU, the state transformation matrix is

$$F = \begin{bmatrix} F_1 & F_2 \\ F_3 & F_4 \\ F_5 & F_6 \end{bmatrix} \in \mathbb{R}^{(3N-7m+24)\times(3N-7m+24)}, \tag{2.59}$$

where each of the blocks in the matrix $F$ is given by

$$F_1 = \begin{bmatrix} I_{3N-7m} & 0 & 0 & 0 & 0 \\ 0 & e^{\omega^{\times}} & 0 & I & \begin{bmatrix} \frac{\partial e^{\omega^{\times}}}{\partial \omega_1} T & \frac{\partial e^{\omega^{\times}}}{\partial \omega_2} T & \frac{\partial e^{\omega^{\times}}}{\partial \omega_3} T \end{bmatrix} \\ 0 & 0 & \frac{\partial \log_{SO(3)}(R)}{\partial R} \frac{\partial R}{\partial \Omega} & 0 & \frac{\partial \log_{SO(3)}(R)}{\partial R} \frac{\partial R}{\partial \omega} \end{bmatrix} \in \mathbb{R}^{(3N-7m+12)\times(3N-7m+12)} \tag{2.60}$$

$$F_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \in \mathbb{R}^{(3N-7m+12)\times(12)} \tag{2.61}$$

$$F_3 = \begin{bmatrix} 0 & 0 & 0 & I_3 & 0 \\ 0 & 0 & 0 & 0 & I_3 \end{bmatrix} \in \mathbb{R}^{6\times(3N-7m+12)} \tag{2.62}$$

$$F_4 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \in \mathbb{R}^{6\times 12} \tag{2.63}$$

$$F_5 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{12\times(3N-7m+12)} \tag{2.64}$$

$$F_6 = [I_{12}] \in \mathbb{R}^{12\times 12} \tag{2.65}$$

and where $R = e^{\omega^\times} e^{\Omega^\times}$, and

$$\frac{\partial(R)}{\partial\Omega} = \begin{bmatrix} \left(e^{\omega^\times}\frac{\partial e^{\Omega^\times}}{\partial\Omega_1}\right)^s & \left(e^{\omega^\times}\frac{\partial e^{\Omega^\times}}{\partial\Omega_2}\right)^s & \left(e^{\omega^\times}\frac{\partial e^{\Omega^\times}}{\partial\Omega_3}\right)^s \end{bmatrix} \tag{2.66}$$

$$\frac{\partial(R)}{\partial\omega} = \begin{bmatrix} \left(\frac{\partial e^{\omega^\times}}{\partial\omega_1}e^{\Omega^\times}\right)^s & \left(\frac{\partial e^{\omega^\times}}{\partial\omega_2}e^{\Omega^\times}\right)^s & \left(\frac{\partial e^{\omega^\times}}{\partial\omega_3}e^{\Omega^\times}\right)^s \end{bmatrix} \tag{2.67}$$

$$\frac{\partial \log_{SO(3)}(R)}{\partial R} = \begin{bmatrix} \frac{\partial \log_{SO(3)}(R)}{\partial r_{11}} & \frac{\partial \log_{SO(3)}(R)}{\partial r_{21}} & \cdots & \frac{\partial \log_{SO(3)}(R)}{\partial r_{33}} \end{bmatrix}. \tag{2.68}$$

Note that the propagation step makes no alterations to the estimate of feature location, nor does it alter the states corresponding to IMU errors, as the covariance of states can only be altered with a measurement. When an IMU estimate is available, the state functions given in Equation 2.58 must be linearized, adding components to the state transformation matrix.

$$F = \begin{bmatrix} F_1 & F_2 \\ F_3^* & F_4^* \\ F_5 & F_6 \end{bmatrix} \in \mathbb{R}^{(3N-7m+24)\times(3N-7m+24)} \tag{2.69}$$

The states blocks corresponding to velocities are altered according to

$$F_3^* = \begin{bmatrix} 0 & 0 & -\begin{bmatrix} \frac{\partial e^{\Omega^\times}}{\partial\Omega_1}\hat{a^B} & \frac{\partial e^{\Omega^\times}}{\partial\Omega_2}\hat{a^B} & \frac{\partial e^{\Omega^\times}}{\partial\Omega_3}\hat{a^B} \end{bmatrix} & I_3 & 0 \\ 0 & 0 & -\begin{bmatrix} \frac{\partial e^{\Omega^\times}}{\partial\Omega_1}\hat{\omega^B} & \frac{\partial e^{\Omega^\times}}{\partial\Omega_2}\hat{\omega^B} & \frac{\partial e^{\Omega^\times}}{\partial\Omega_3}\hat{\omega^B} \end{bmatrix} & 0 & I_3 \end{bmatrix} \in \mathbb{R}^{6\times(3N-7m+12)} \tag{2.70}$$

$$F_4^* = \begin{bmatrix} -\left(e^{\Omega^\times}\right)^T D_a & -\left(e^{\Omega^\times}\right)^T & 0 & 0 \\ 0 & 0 & -\left(e^{\Omega^\times}\right)^T D_g & -\left(e^{\Omega^\times}\right)^T \end{bmatrix} \in \mathbb{R}^{6\times 12}. \quad (2.71)$$

The covariance is propagated according to

$$P\left(t+1|t\right) = F\left(t\right) P\left(t|t\right) F^T\left(t|t\right) + \Sigma_w\left(t\right). \quad (2.72)$$

## 2.6 Implementation Details

### 2.6.1 Observability

Following the derivations in *An Invitation to 3-D Vision* it can be shown that the model described in this section is observable up to a similarity transformation, provided that the translational velocity is nonzero [1]. This means that there are multiple trajectories that would produce the same measurements as the true motion. These trajectories each belong to an equivalence class that is indistinguishable from the true trajectory. If the dimension of the equivalence class can be reduced to one, then only the true trajectory would remain and the system would be observable. To render the system observable, the directions of three features and the depth (scale factor) of one feature are fixed. These elements are sufficient to identify the true equivalence class, and hence the true trajectory. [1], [10] In practice the reference states are chosen to represent the features with the greatest confidence on the depth estimates. The reference states are removed from the state vector, as they are not quantities to be estimated, and are maintained as reference constants.

### 2.6.2 Necessary Number of Features

The extended Kalman filter described is based on the theory of epipolar geometry, presented in Section 2.2. To ensure that the essential matrix, Equation 2.14, is full rank, and that there is a unique solution, there must be a minimum of eight features

matched in a pair of images. However, in a filtering architecture there need not necessarily be eight features tracked in every frame. On average there must be at least eight features in order to calculate an accurate motion estimate. If in a few frames the feature count drops below eight the filter will not immediately diverge. In practice it is useful to track more than the minimum number of features because as the vehicle moves into a new environment some features will be lost. Although it may be possible for the filter to operate with fewer than eight features, it is always desirable to have more measurements.

### 2.6.3 Handling Lost Features

When a feature moves out of the field of view of the camera or becomes occluded it is considered lost. In this case the feature is removed from the state and the covariance. Maintaining a feature that has no new measurements is computationally wasteful and could result in ill-conditioned inverses. The states $x_0^i(t)$ and $\lambda^i(t)$ are removed from the state vector and the corresponding rows and columns are removed from the covariance matrix. It is acceptable to simply remove them from the covariance matrix because the states are decoupled.

In practice it is important to ensure not only that the lost features have been removed from the state vector, but that the new measurements are arranged in the correct order and that no additional measurements are present.

#### When A Lost Feature Causes Drift

The only time that losing a feature causes a problem is if that feature is one of the three features that is used to fix the observable component of the state space ($i = 1, 2, 3$). When this happens, the feature indices must be shuffled such that the reference features are filled. That is, if feature $i$ is lost at time $\tau$, and $i < 4$, then $x_0^i$ and $\lambda^i$ must be replaced by one of the persisting features $j > 3$. To do this, the feature information from $j$ is placed into the proper locations for feature $i$ and then feature $j$ is removed, as if it were lost.

In the case where $x_0^j(\tau)$ and $\lambda^j(\tau)$ are precisely equal to the true values, shifting the reference states to feature $j$ will have no effect on the accuracy of the egomotion estimate. However, in general this will not be the case. The difference $\tilde{x}_0^j(\tau) \doteq x_0^j(\tau) - \hat{x}_0^j(\tau)$ is a random variable with covariance $\Sigma_x$, which is available from the corresponding block of $P(\tau|\tau)$. Since the filter estimates motion by comparing the measured position of a feature to its expected position (propagated from its initial position), and discrepancy in initial position would result in an error in the motion estimate. Thus, switching the reference to feature $j$ causes a drift proportional to $\tilde{x}_0^j(\tau)$, which has a standard deviation of $\sqrt{\Sigma_x}$. And after making $M$ switches of any of the three reference features, it is expected that the drift due to feature position uncertainty will be proportional to $M\left\|\sqrt{\Sigma_x}\right\|$.

An additional source of drift occurs when the first reference feature is lost (the one that also has a reference depth). Recall that to solve the scale factor ambiguity it was necessary to fix the depth estimate to one feature, upon which all other state estimates are based. If this feature is lost, a new feature must be chosen to be the dominant reference. In general, this new feature depth estimate will not be exact, and will have a covariance $\Sigma_\tau$. An error in depth will manifest itself as an error in the translation estimate. An uncertainty on the reference depth, given by

$$\lambda^1 = \lambda^1_{\text{Truth}} + \delta\lambda, \tag{2.73}$$

where $\delta\lambda$ is the standard deviation $\left(\sqrt{\Sigma_\tau}\right)$, will result in a translation uncertainty of

$$\tilde{T} = \begin{bmatrix} \delta T \\ \delta T \\ \delta\lambda \end{bmatrix}. \tag{2.74}$$

$\delta T$ is the error in position resulting from an unknown depth and is proportional to $\delta\lambda$. The magnitude of the translation error (drift) resulting from uncertain depth estimates is

$$||\tilde{T}|| = \sqrt{2(\delta T)^2 + (\delta\lambda)^2}. \tag{2.75}$$

And since $\delta T \propto \delta\lambda$,

$$||\tilde{T}|| \propto \delta\lambda. \tag{2.76}$$

After making N switches of the primary reference feature, it is expected that the drift due to depth uncertainty will be proportional to $N||\sqrt{\Sigma_\tau}||$. Since the depth uncertainty is much larger than the feature position uncertainty, this is the dominant source of drift.

This is an unavoidable consequence of losing the reference features. To minimize the effect of switching features, it is important to choose the new feature $j$ that has the lowest covariance. In practice, if $i = 1$ is lost it is replaced by either feature 2 or feature 3 (whichever has the lower covariance on depth). If $i = 2, 3$ is lost, or if $i = 1$ is lost and it is replaced by 2 or 3, then the lost feature is simply replaced by the feature with the lowest covariance on depth. [10] To further reduce the effect of switching features the cameras may be oriented in the direction of greatest feature persistence.

### Recovering Previous Drift

In the case where a reference feature has been lost it may be beneficial to store that feature information. If that feature should reappear in a future frame, the reference state could be switched back to the original feature. This would recover any drift that had accumulated in the interim. [8], [17] This feature has not been included in this implementation, although from the Kalman filter side it would add little complexity. All that would need to be stored is the $x_0$ and $\lambda$ values for a lost reference features, as well as the feature identification tag. This is only three doubles and one integer, a fairly low computational burden. These values need not be updated during the time the feature is lost.

The greatest challenge to implementing this capability comes about in the feature identification and tracking algorithms. The identifying pixel region would need to be stored for every feature and all new features coming into the image would need to be compared to the database of previously viewed features. As the number of previously

viewed features increases, this comparison would quickly become unreasonable. This problem transforms from one of visual odometry to SLAM, and requires building a map of the entire observed world. Additionally, if the environment changes during the unobserved period (for instance, a car moves position), the filter would not recover its lost drift as desired. Rather, it would gain additional drift by the amount the reference feature has moved. For this application it is likely unreasonable to implement this feature, however it would be an interesting way to recover drift in a stationary environment.

### 2.6.4 Handling New Features

When a new feature becomes available it is important to track this feature for a few frames before incorporating it into the state vector. Recall that the feature information that is included in the state vector is the feature's location at time $t = 0$. Since the new feature becomes available at time $t = \tau > 0$, it is impossible to determine the information at $t = 0$ without having some more information on the feature. Especially important to determine is the distance to the feature, which must be accomplished through filter convergence. The process of tracking a new features is accomplished through a subfilter which runs in parallel with the main filter.

An independent subfilter is maintained for each feature visible that is not tracked by the main filter. The basic framework of the subfilter is the same as that of the main filter. The greatest difference is that the subfilter is not estimating the position of the feature at time $t = 0$ with the terms $x_0^i(t)$ and $\lambda_i(t)$. Rather, the subfilter is estimating the position of the feature at time $t = \tau$, the first time at which the feature is observed.

#### Subfilter Structure

The subfilter state vector is given by

$$\hat{\mathbf{x}} = \begin{pmatrix} \hat{\mathbf{x}}_\tau(t) \\ \hat{\lambda}_\tau(t) \end{pmatrix} \in \mathbb{R}^{3x1} \tag{2.77}$$

where $\hat{\mathbf{x}}_\tau(t)$ is the estimate of the two dimensional feature location at time $t = \tau$ and $\hat{\lambda}_\tau(t)$ is the estimate of the depth at that time.

### Subfilter Initialization

The state vector is initialized to be the best estimate of the feature location. The elements corresponding to $\hat{\mathbf{x}}_\tau(\tau|\tau)$ are set to the observed feature location, $\mathbf{x}(\tau)$. The depth estimate $\hat{\lambda}_\tau(\tau|\tau)$ can simply be set to 1. The state error covariance matrix is initialized to

$$P_\tau(\tau|\tau) = \begin{bmatrix} \Sigma_n(\tau) & 0 \\ 0 & M \end{bmatrix} \in \mathbb{R}^{3x3} \tag{2.78}$$

where $\Sigma_n$ is the measurement noise covariance and is an estimate of the confidence in the observed feature location. $M$ is a large positive number, chosen to represent an initial lack of confidence in the depth estimate.

### Subfilter Prediction Step

The subfilter is propagated similarly to the main filter, with the distinction that the state transformation matrix $F$ is identity.

$$\begin{aligned} \hat{\mathbf{x}}_\tau(t+1|t) &= \hat{\mathbf{x}}_\tau(t|t) \\ \hat{\lambda}_\tau(t+1|t) &= \hat{\lambda}_\tau(t|t) \\ P_\tau(t+1|t) &= P_\tau(t|t) + \Sigma_w(t) \end{aligned} \tag{2.79}$$

### Subfilter Update Step

As expected, the update step is very similar to that of the main filter. The main difference is that the feature location estimate must be transformed back to the estimated location at time $t = 0$.

$$\hat{\mathbf{x}}_0(t+1|t)\,\hat{\lambda}_0(t+1|t) = e^{R_{\text{Cam}}^T \Omega^\times(\tau)} \left[\hat{\mathbf{x}}_\tau(t+1|t)\,\hat{\lambda}_\tau(t+1|t) + R_{\text{Cam}}^T T(\tau)\right] \tag{2.80}$$

The estimates of translation and rotation are taken directly from the main filter and are assumed to be accurate. So the update equation is given by

$$\begin{bmatrix} \hat{\mathbf{x}}_\tau(t+1|t+1) \\ \hat{\lambda}_\tau(t+1|t+1) \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{x}}_\tau(t+1|t) \\ \hat{\lambda}_\tau(t+1|t) \end{bmatrix} + L_\tau(t+1) \times \left\{ \mathbf{x}(t+1) - \pi\left( e^{-R_{\text{Cam}}^T \Omega^\times (t+1)} \hat{\mathbf{x}}_0(t+1|t)\, \hat{\lambda}_0(t+1|t) - R_{\text{Cam}}^T T(t+1) \right) \right\} \tag{2.81}$$

**Adding New Features To The Main Filter**

As the subfilter runs for each feature, the depth estimate will slowly converge to the true value. As this happens, the covariance will shrink. Once the covariance has reached a specified threshold, the feature is considered ready to be added to the main filter. The initial states at time $t = 0$ are calculated using Equation 2.80 and are inserted into the main state vector. They are simply added in the appropriate places in the state vector and the corresponding matrices are adjusted as well. The covariance of the new feature is set to the covariance of the estimation error in the subfilter.

### 2.6.5 Initialization

A number of different methods have been explored for initializing the state vector. The feature location on the image plane, the IMU parameters, and the position and attitude vectors are trivial to initialize in the state vector. The subvector that requires substantial work is the feature depths. This is information that is not readily available. Two approaches have been implemented to initialize the depths. The first is used for testing only and involves initializing the depths to the correct values. This is useful in simulation, laboratory experiments, and when additional information is available because it provides a benchmark for the error introduced through feature depth uncertainty.

The second approach involves estimating the depths using assumed truthful mo-

tion estimates from the GPS and IMU. The subfilter described in section 2.6.4 is used in this initialization procedure. The subfilter is run for a few seconds using estimates from the GPS and IMU as position and attitude information to estimate the depths of each feature. Then the depth estimates are inserted into the new filter, with the lowest covariance features serving as the reference features.

The initial covariance matrix is chosen to be diagonal with the following structure. The diagonal elements corresponding to $x_0$ are set to $\sigma_{\text{measurement}}$, those corresponding to $\lambda$ are initialized to $M$, those corresponding to $T$ and $\Omega$ are set to zero, the initial velocity covariances are set to $W$, and the initial covariances on the IMU biases and scale factor errors are set to $U$. $\sigma_{\text{measurement}}$, $M$, $W$, and $U$ are all filter parameters that are tuned to achieve the best filter performance.

### 2.6.6 Tuning Filter Parameters

A number of parameters must be properly tuned to achieve the desired performance of the extended Kalman filter. Tuning nonlinear filters is an art and the tuning suggestions provided here are meant to serve only as a starting point. The values given have worked for simulations and preliminary live-camera experimentation, however a deployable system may require different values. The filter parameters are divided into two categories - initialization and propagation parameters.

The initialization parameters are used for initializing the covariance matrix, as described in Section 2.6.5. $M$ and $W$ are chosen to be large positive numbers. They correspond to the initial covariance on feature depth and camera velocity, respectively, two quantities that are not known with much certainty during initialization. Using values of 100 for both $M$ and $W$ works well. The parameter $U$ refers to the initial covariance on the IMU biases and scale factor errors. For simulations this has been set to 1.

The propagation parameters are used for propagating the filter forward in time, after the initialization is complete. $\sigma_{\text{measurement}}$ is the covariance of the measurement and is included in the covariance initialization, the model error covariance, and the measurement error covariance. This quantity is related to the performance of the

feature tracking algorithm and refers to the uncertainty on the measurement position. A standard deviation of one pixel may be a reasonable starting value. Realizing that one pixel of a 640 pixel image is 0.0016 of the visible image, the value of $\sigma_{\text{measurement}}$ has been set to 0.001. $\sigma_\lambda$ is the model error covariance on the depth estimate and $\sigma_v$ is the model error covariance on velocity. *An Invitation To 3D Vision* recommends changing $\sigma_v$ relative to $\sigma_\lambda$ to allow for more or less regular motion and changing both relative to $\sigma_{\text{measurement}}$ to adjust the level of desired smoothness in the estimates [10]. In practice, $\sigma_\lambda$ has been chosen relatively large, 100, to allow the filter to rapidly converge on the depth estimate. $\sigma_v$ has been chosen much smaller, 0.01, to discourage confusion between translation and the corresponding rotation direction.

# Chapter 3

# Investigation Of The Contribution Of Multiple Cameras

Using the extended Kalman filter developed in the previous section, the benefits of using multiple cameras for egomotion estimation are examined here. To decouple the effects of poor image processing and feature tracking from the workings of the Kalman filter, a number of computer simulations were performed. To approximate the performance of a real camera, the angular field of view is chosen to be $\pm 22^{\circ}$ in the horizontal direction and $\pm 17.4^{\circ}$ in the vertical direction. These simulations were performed in Matlab and involve a simulated feature track, represented by the screenshot in Figure 3-1. The different colors represent screenshots at one second intervals as the features move from the top of the frame to the bottom. The features are randomly distributed in three dimensional space and the simulated camera is maneuvered through the feature cloud along a know trajectory. Note that not all features move with the same apparent velocity, this is caused by the features being at different distances from the simulated camera. With this method of simulation all features in the field of view are measured perfectly with no feature confusion possible. With perfect feature tracks it is then possible to test the performance of the filter alone. This serves as a bound on performance when true image processing is performed.

The goal of this section is to test the Kalman filter in a variety of flight regimes

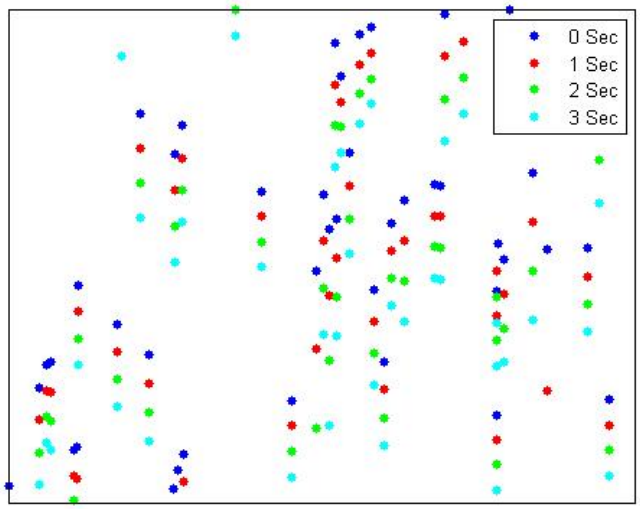


Figure 3-1: Simulated feature track

and evaluate the performance in each. Recommendations will be made about the necessary number of cameras and the suggested orientations. It should be noted that these are minimal recommendations and, when computational capabilities allow, adding more cameras will not compromise the estimates obtained using a single camera and IMU. Additionally, some interesting phenomena have been explored in simulation and will be presented.

## 3.1 Flight Regimes

For this study flight profiles are divided up based on how long the features remain in the field of view. The greatest feature persistence is demonstrated in the case of an aircraft loitering over a stationary target. Although the aircraft will undergo translational and rotational motion during its observation period, for this study it is assumed that all or most of the features remain in the field of view indefinitely. Next in terms of feature persistence is a high altitude and/or a low velocity trajectory. The pixel velocity of features is low in this type of flight and the features remain in the field of view for a relatively long period of time. As velocity increases and altitude

decreases, the features get closer to the camera and the pixel velocity increases. This means that the features will persist for a shorter period of time. The extreme situation would be where the aircraft is flying at a very low altitude at a high velocity while trying to estimate motion using a downward looking camera. This sort of flight profile is similar to flight through an urban canyon (flying down a street below rooftop level).

### 3.1.1 Performance Metrics

The relevant performance metric to evaluate the quality of the vision based egomotion estimation system is the magnitude of the estimation error. In computer simulations it is relatively easy to create error estimates because the true trajectory of the vehicle is known. Unlike with an inertial estimate, the estimation error for a vision based system does not grow with time. Rather, the estimation error grows with distance traversed. As long as a majority of the original features remain in the field of view of the camera, the system will not accumulate substantial drift. It is only when the reference features that have strong depth estimates leave the field of view and new features enter does drift accumulate.

## 3.2 Translation-Rotation Ambiguity

The physics of a projective camera impose an ambiguity on the motion estimate in certain situations. In a situation with sufficiently small motions, and with a narrow field of view, there is an ambiguity between rotation and translation when using a single camera for motion estimation. This ambiguity is caused by a loss of observability, as described in Section 2.6.1. Consider a set of planar features located at a unity depth (for simplicity) and contained in the camera's field of view.

$$\mathbf{X} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.1}$$

Recall that translation and rotation alter the feature's position on the image plane according to

$$\mathbf{x} = \pi\left(R\mathbf{X} - T\right) \tag{3.2}$$

If the camera undergoes a small translation in the $x$ direction $T_1 = \begin{bmatrix} \Delta x & 0 & 0 \end{bmatrix}^T$ and no rotation ($R_1 = I$), the observed feature position on the image plane will be given by

$$\mathbf{x}_1 = \pi\left(R_1\mathbf{X} - T_1\right) = \pi\left(\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} - \begin{bmatrix} \Delta x \\ 0 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} x - \Delta x \\ y \end{bmatrix} \tag{3.3}$$

Now consider the case where the camera undergoes no translation $T_2 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$ and a rotation about the $y$ axis by a small negative angle, $\Omega_2 = \begin{bmatrix} 0 & \Delta\phi & 0 \end{bmatrix}^T$ and $R = e^{\Omega^\times}$. For sufficiently small $\Delta\phi$ the rotated feature coordinates on the image plane are given by

$$\mathbf{x}_2 = \pi\left(R_2 * \mathbf{X}\right) \approx \pi\left(\begin{bmatrix} x + \Delta\phi \\ y \\ 1 - \Delta\phi \end{bmatrix}\right) = \begin{bmatrix} \frac{x+\Delta\phi}{1-\Delta\phi} \\ \frac{y}{1-\Delta\phi} \end{bmatrix} \tag{3.4}$$

For small translations, $\Delta\phi$ may be chosen such that $\mathbf{x}_1 \approx \mathbf{x}_2 + \sigma_{Measurement}$. That is, the two cases may be indistinguishable when the measurement noise is taken into account.

These unobservable modes exist in the cases of translation in the camera's $x - y$ plane and rotation about the $x$ and $y$ axes. This effect may be reduced when the features are further away from the optical axis and when the features exist at diverse distances. Using a camera with a wide field of view helps to reduce this ambiguity. Since this ambiguity becomes apparent in many single camera flight regimes, a brief demonstration of the ambiguity as well as some suggestions for mitigating it are included here.

### 3.2.1 Ambiguity Caused By Small Motion

To demonstrate how ambiguity can be caused by small magnitudes of motion, the following simulation was performed. The simulation consists of a view that is stationary for two seconds, followed by a forward acceleration. Motion in all other translation and rotation directions is zero. The forward motion of the vehicle is shown in Figure 3-2. The total motion of the vehicle over ten seconds is about 32 meters at an altitude of approximately 150 meters.

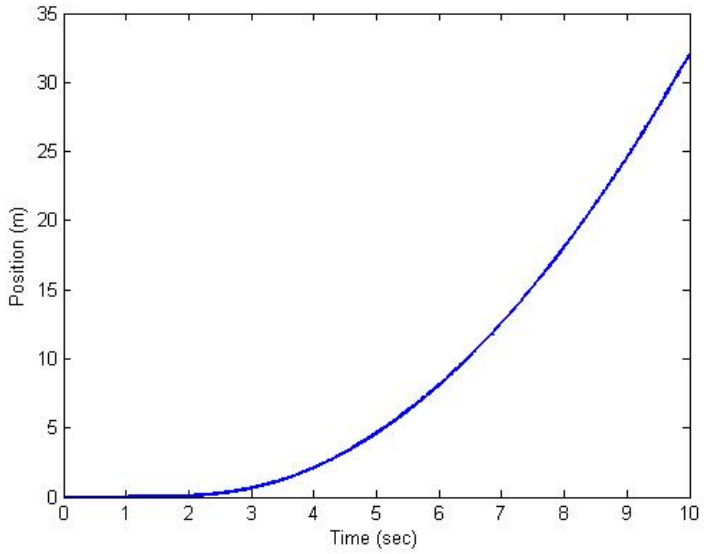


Figure 3-2: Translation-rotation ambiguity simulation trajectory

### Single Camera

When a single downward looking camera is used to estimate this motion there is confusion about the type of motion. Figure 3-3 shows the estimated motion plotted with the true trajectory. Note that the filter estimates the vehicle has moved less than it actually has. For clarity Figure 3-3 is expressed as an error plot in Figure 3-4.

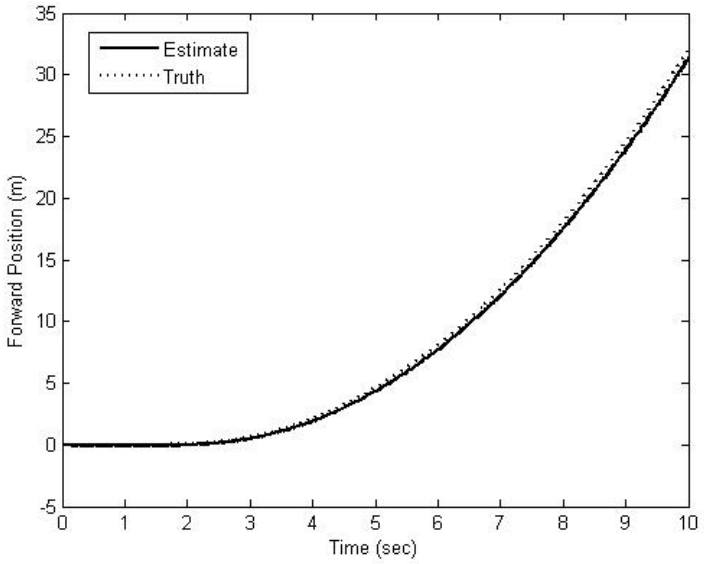


Figure 3-3: Single camera simulation results - forward motion

It is clear from Figure 3-4 that the error appears as the vehicle accelerates from a stationary position at 2 seconds. As expected for the small magnitude of translation in the first few seconds, the observability condition of nonzero velocity is violated and the vehicle pitch is estimated to be larger than the true value. A decreased translation value indicates that the vehicle did not travel far enough forward. To obtain the same measurement of feature position, the filter must estimate that the vehicle is pitching up. Figure 3-5 shows the true vehicle pitch (identically zero) and the estimated vehicle pitch (nonzero). An estimated pitch occurs when the vehicle begins moving forward. This positive pitch accounts for the decreased translation estimate. All translation and rotation directions not shown closely match the true zero values.

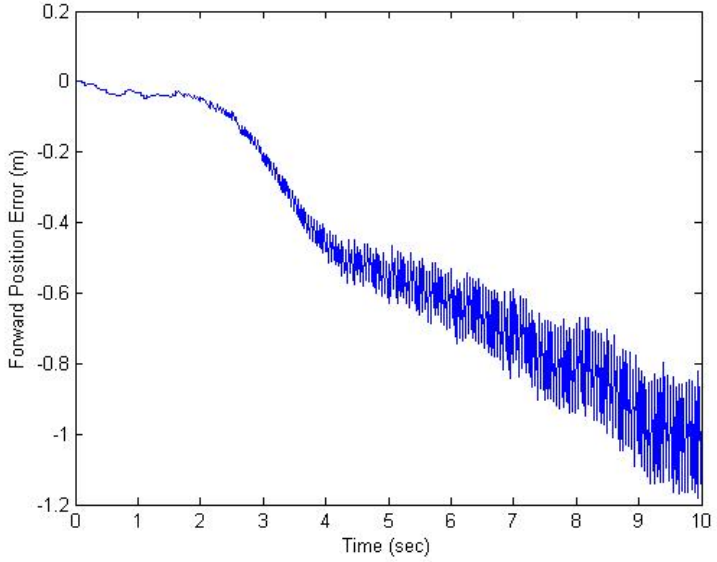


Figure 3-4: Single camera simulation results - forward error

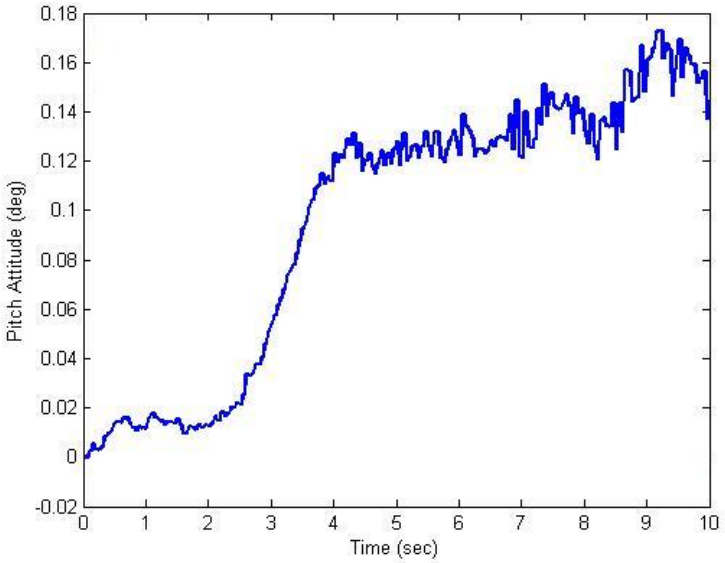


Figure 3-5: Single camera simulation results - pitch error

Since the velocity is initially zero, the system starts out unobservable and a translation-rotation ambiguity results. Once the estimate becomes anchored in a certain motion equivalence class, observability is recovered. However, the motion it is anchored in could be the incorrect trajectory. For a single camera, in which the z-axis is oriented along the centerline of the camera, rotation about the z-axis or translation in the direction of the z-axis are unambiguous. Translation in any other direction or rotation about any other axis can be ambiguous. It may be stated that an error of less than a tenth of a degree is acceptable, and in general this is true. Attitude estimates with error less than one degree are considered excellent for this type of sensor. However, in a different experiment when the depths to the features are less well known the opportunity for confusion is much greater and a larger degree of error can be expected. Perhaps more significantly, this ambiguity results in a large error in translation estimation that continues to grow with time. This divergence begins at first movement and continues to grow throughout the simulation. In a long distance experiment, this ambiguity will cause increased translation error

### 3.2.2 Ambiguity Caused By Clustered Features

Translation-rotation ambiguity can also be caused by a lack of linearly independent feature measurements. In the case where all of the features are clustered in a small fraction of the image plane, the features do not all contribute additional useful information. To quantify the clustering of the features, the area of the smallest bounding box enclosing all features is compared to the area of the entire image. For instance, a 1% clustering indicates that all visible features are enclosed within an area $\frac{1}{100}$ the area of the image. In this situation additional features do not provide much more information than a single feature in that location, they are linearly dependent. To quantitatively assess the relative value of different cluster levels, the *Fisher Information Matrix* (FIM) is calculated [19]. This matrix is a measure of how much information about the vehicle's motion is contained in a set of measurements. It is related to the performance of an optimal, unbiased estimator. The Cramer-Rao bound states that the inverse of the determinant of the information matrix is a lower

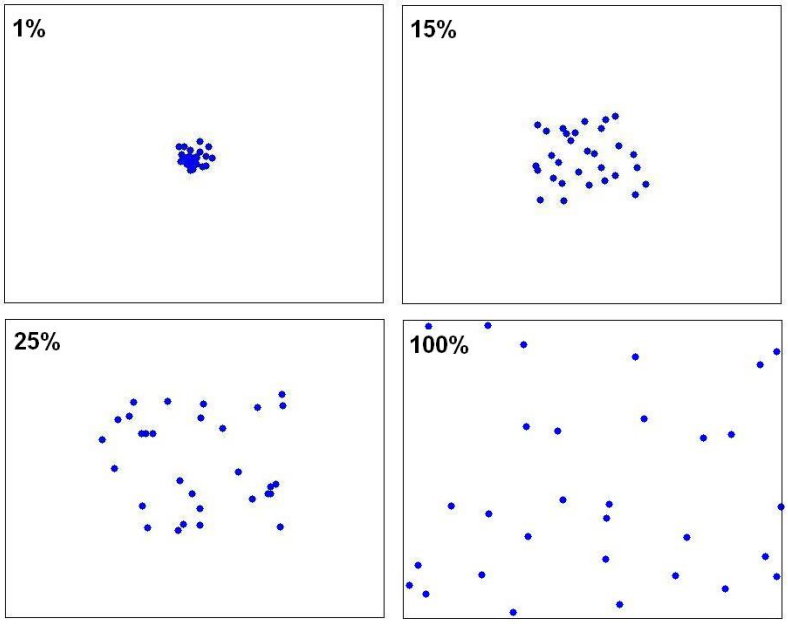


Figure 3-6: Four clustered feature levels

bound on the covariance [12]. The FIM, $J$, is given by

$$J = H_x^T \Sigma_n^{-1} H_x \tag{3.5}$$

where $H_x$ is the ideal measurement sensitivity matrix in the Kalman filter, which contains information on how each state is affected by a set of measurements. The ideal measurement sensitivity matrix is similar to the measurement sensitivity matrix, with the distinction that it is formed by linearizing the measurement equation about the true state $x$, not the estimated state $\hat{x}$.

$$H_x = \frac{\partial h_{\text{Vision}}}{\partial x} \tag{3.6}$$

Since we are only interested in determining how much information about the translation and rotation states is present, and not necessarily about all of the other states, the $H$ matrix is reduced to only the six columns that relate to these states. So $J$ is a $6 \times 6$ matrix.

An information matrix that is relatively "large" indicates a great deal of information about the camera's motion is provided by the set of measurements. A relatively "small" information matrix indicates a lack of information, likely caused by a set of linearly dependent measurements. The size of a matrix is evaluated by the determinant. An information matrix with a larger determinant will contain more information than one with a small determinant. A series of simulations were performed with various feature clustering levels and the determinants of the information matrices were recorded. Additionally, for each level of clustering the simulation was performed using one camera, two orthogonal cameras, and three orthogonal cameras. The results for a case where there were 30 features visible in each camera are shown in Figure 3-7. This plot demonstrates that for each camera configuration there is more information present when the measurements are dispersed evenly about the entire image. It also demonstrates that more information about the vehicle's motion is available when multiple cameras are used.

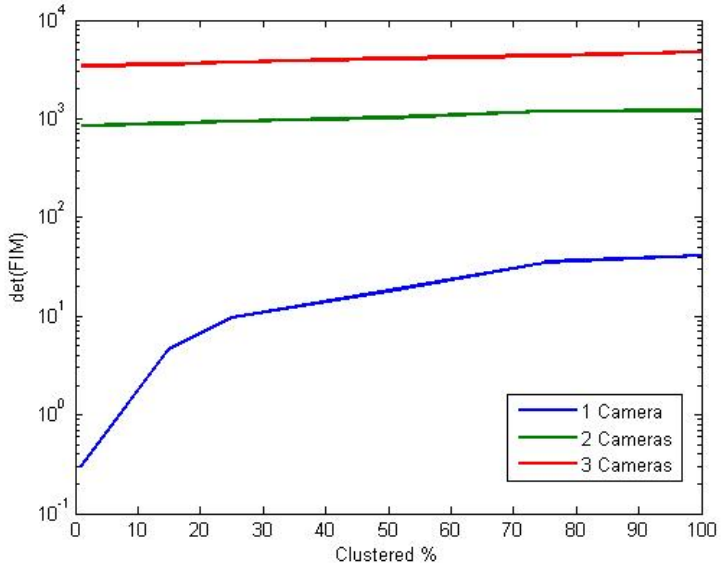


Figure 3-7: Fisher Information Matrix results - 30 features

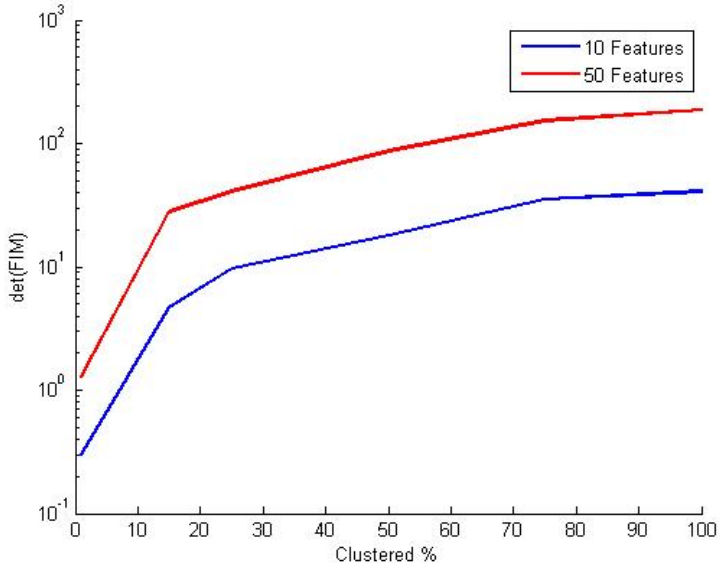


Figure 3-8: Fisher Information Matrix results - feature number comparison

The Fisher Information Matrix is also useful for evaluating how much information is contained in a set of measurements when different numbers of features are visible. A simulation was performed in which a single camera was used to observe environments with a small number of features (10) and a large number of features (50). As expected, Figure 3-8 indicates that more features correspond to more information. An interesting result is that more information can be achieved when using a small number of unclustered features than when using a large number of clustered features. This result confirms that operating with substantial feature distribution is more desirable than simply tracking a large number of features. This result can be extended to suggest that a larger field of view is desirable, as it allows the features to be distributed further from the camera center.

A simulation was performed to assess the impact of using multiple cameras where one of the cameras does not have many features in the field of view. In this simulation a configuration of two orthogonal cameras is used. One camera tracks 30 features, while the other can only track 4. The results are shown in Figure 3-9, plotted with the same results as Figure 3-7. The result '2 Limited Cameras' correspond to this case where one of the two cameras has a limited number of features. Clearly, the information provided by these four features is superior to the information obtained using a single camera. Despite the fact that there are only a few features in the image, useful information is provided to the filter.

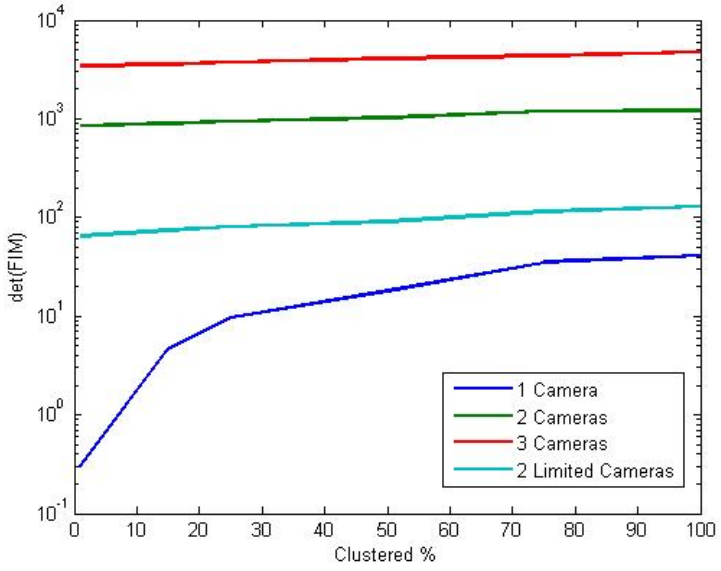


Figure 3-9: Fisher Information Matrix results - few features

These results clearly indicate an increasing amount of information in the set of measurements when the features are more widely distributed. To demonstrate the effect that feature clustering can have on the motion estimates, a simple test case was simulated for four different levels (1%, 15%, 25%, and 100%). A screenshot of each level is shown in Figure 3-6. The flight trajectory is a simple forward translation at a constant rate of ten meters per second for five seconds. To isolate the effects of feature clustering the simulations are performed using a single camera without the help of GPS or an IMU and the feature depths are initialized to the true values.

### 100% Feature Distribution

As a performance benchmark for comparison, the first simulation presented has features distributed over 100% of the image. The motion estimate errors over five seconds, and after traveling fifty meters at an altitude of 200 meters, are shown in Figure 3-10. As expected for such a simple motion over a short distance, there is essentially no translation-rotation ambiguity in any axis, all attitude errors are less than 0.03°.

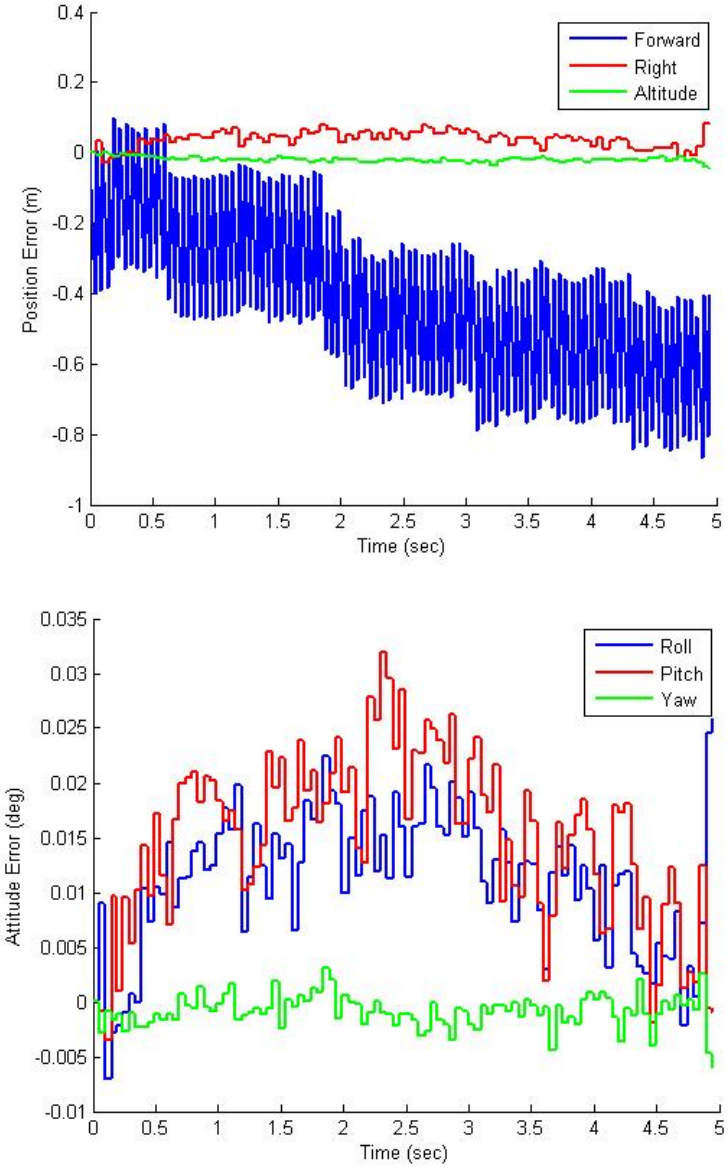


Figure 3-10: 100% distributed features results

### 1% Feature Distribution

Demonstrating the other extreme of feature clustering, this simulation has a feature distribution of only 1%. As can be seen in the top left plot of Figure 3-6, this situation is essentially one large feature. The motion estimate errors are shown in Figure 3-11. In this case there is a great deal of translation-rotation ambiguity in both pitch and roll, as much as 10° pitch error and 35 meters position error after only five seconds.

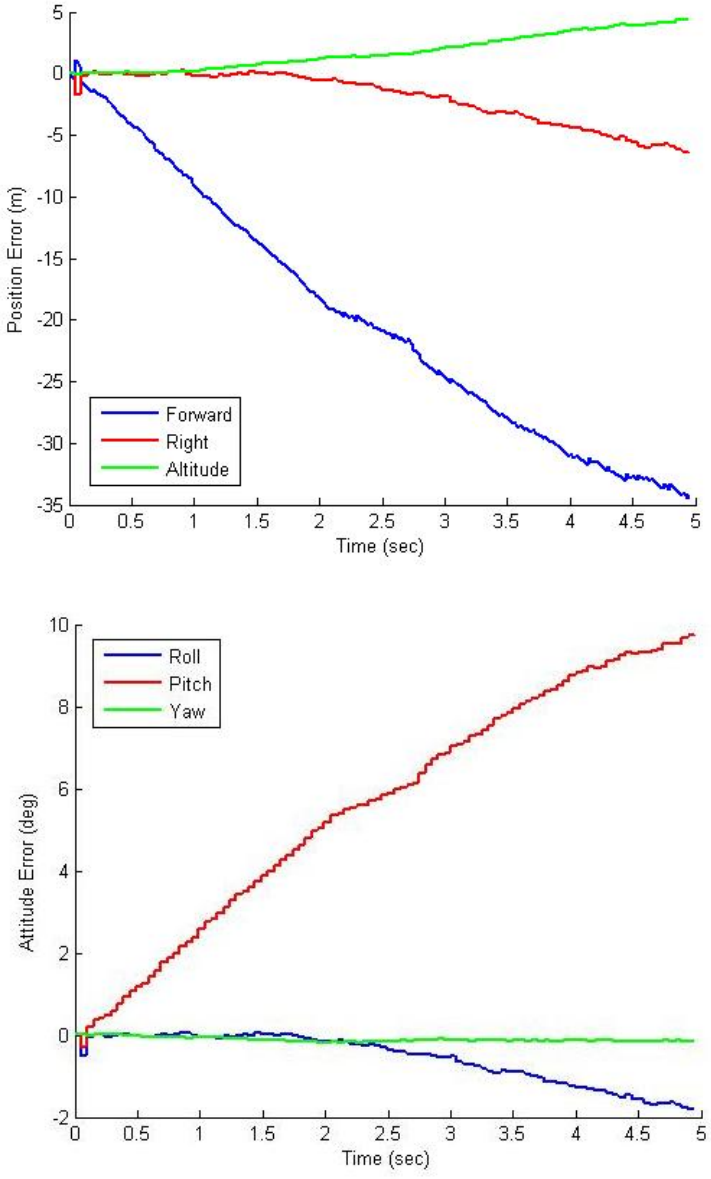


Figure 3-11: 1% distributed features results

### 15% Feature Distribution

It can be easily imagined that the top right plot in Figure 3-6 is a top-down view of a small building in an otherwise featureless world. The motion estimate errors are shown in Figure 3-12 and are significantly improved over the estimates for the 1% case. Although it is approaching a reasonable level, there is still some translation-rotation ambiguity that will continue to grow in a longer simulation.

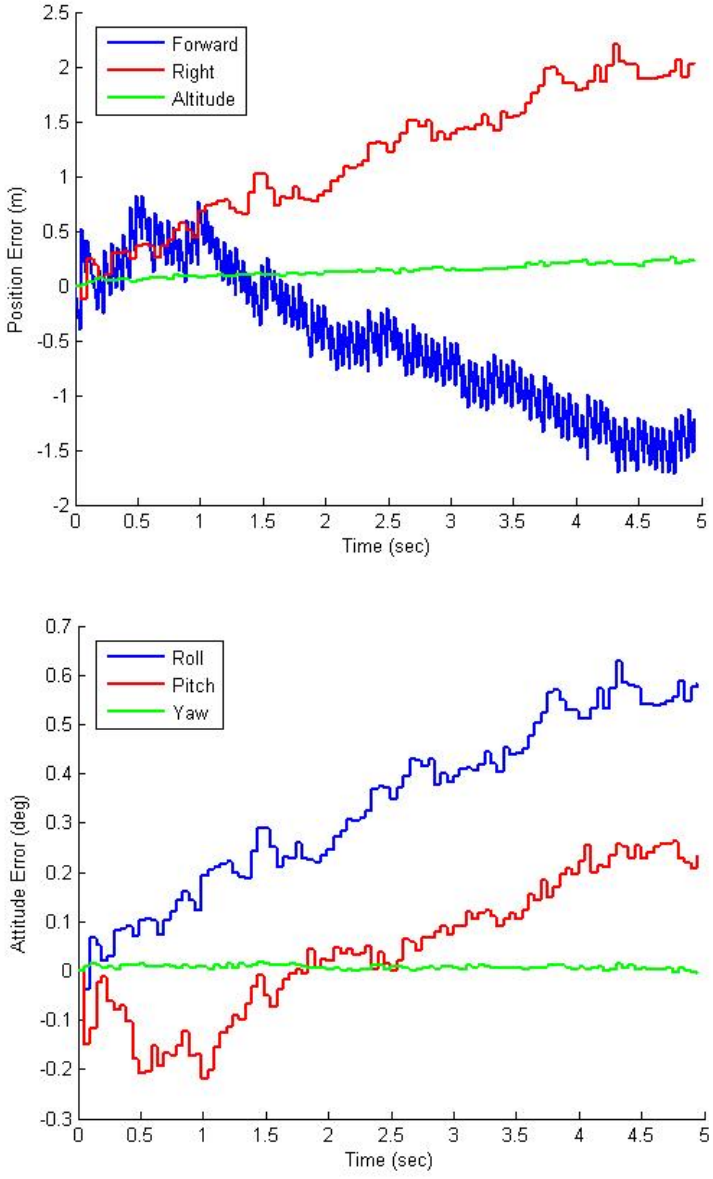


Figure 3-12: 15% distributed features results

### 25% Feature Distribution

25% distribution is a reasonable approximation of motion in a fairly feature sparse world. If there are a few dominant objects in the center of the screen, a standard feature tracker could produce an image similar to the lower left plot in Figure 3-6. The motion estimates are shown in Figure 3-13 and are almost as accurate as the 100% case. There is only a small amount of ambiguity in the forward/pitch direction.

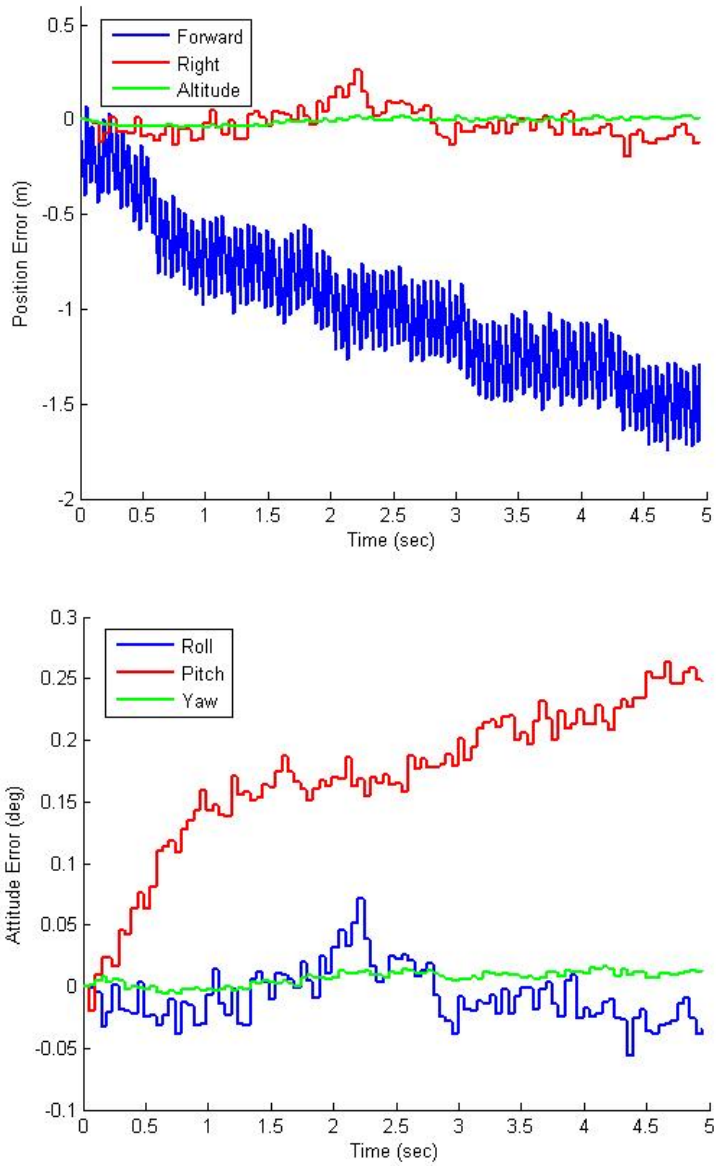


Figure 3-13: 25% distributed features results

### 3.2.3 Reducing Ambiguity Using Multiple Cameras

Based on the Fisher Information Matrix calculations, this is a situation where integrating measurements from multiple cameras is extremely helpful. In the previous sections it was determined that using a downward looking camera alone produces a confusion between forward motion and pitch. The worst case is found to be when there is significant feature clustering. Another simulation is performed with a rightward looking camera viewing another 1% distributed region. The estimate is similarly poor, with the expected confusion between forward motion and yaw. The magnitude of these errors is comparable to that obtained with the downward looking camera. A simulation using both the downward looking camera and the lateral camera was performed to determine if these complementary measurements could work together to estimate the true motion of the aircraft. The two cameras are observing equally sparse regions, yet we know from the FIM that together they can create a valid motion estimate. The results are shown in Figure 3-14.

The results demonstrate that using multiple cameras significantly reduces the likelihood of translation-rotation ambiguity. The position errors are reduced by a factor of fifty at least and the attitude errors are reduced by over two orders of magnitude. Using multiple cameras clearly reduces the translation-rotation ambiguity caused by a lack of linearly independent feature measurements to an acceptable level.

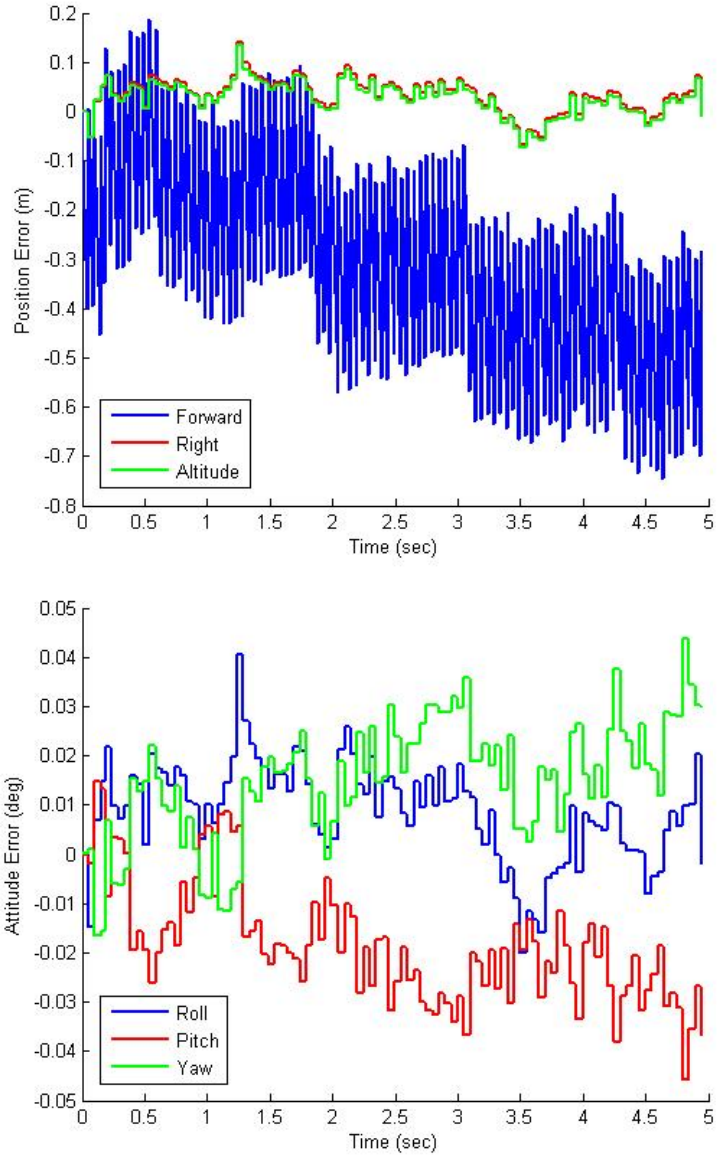


Figure 3-14: Two orthogonal cameras with 1% feature distribution results

## 3.3 Indefinite Feature Persistence

To demonstrate that drift does not accumulate with time a simulation is performed in which the same features stay in the field of view for an extended period of time. This sort of flight path could be performed by a helicopter that is maneuvered to observe a stationary ground target. One dimension of sinusoidal translation and one axis of sinusoidal rotation were chosen for simplicity, however a similar case could include a vehicle circling a stationary target. This sinusoidal motion results in features moving in a combination of top to bottom motion and circular motion on the image plane, which enables the camera to track the same features indefinitely. The features are initialized using GPS for the first three seconds, then the GPS signal is disabled. Excerpts from the vehicle's 10 minute trajectory are expanded for greater clarity in Figures 3-15 and 3-16.

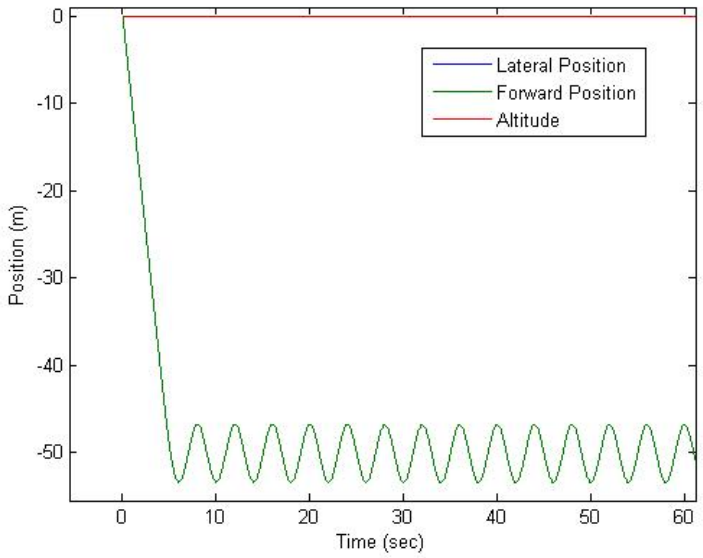


Figure 3-15: Position trajectory for excerpt of 10 minute simulation

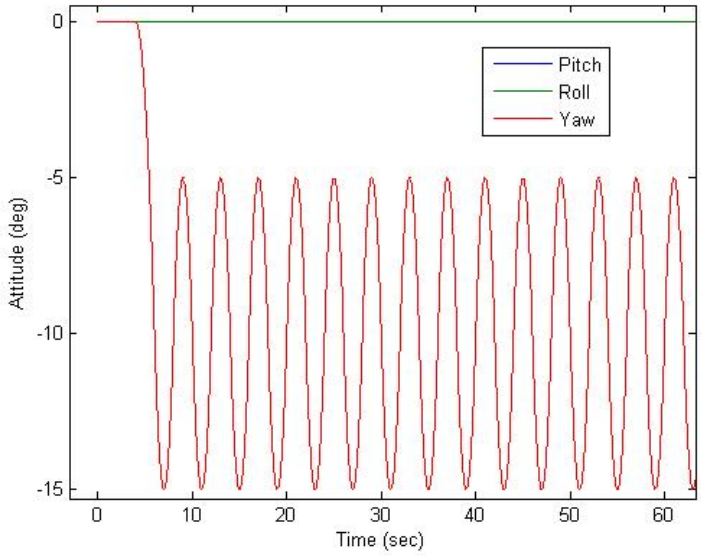


Figure 3-16: Attitude trajectory for excerpt of 10 minute simulation

### Results

The simulation was first performed using a single, downward looking camera. The results demonstrate that the vision based Kalman filter solution is stable for extended periods of time and that the estimation errors do not grow without bound. Over a 10 minute period an IMU operating alone would have unacceptable levels of error. The position and attitude errors in all directions are shown in Figure 3-17.

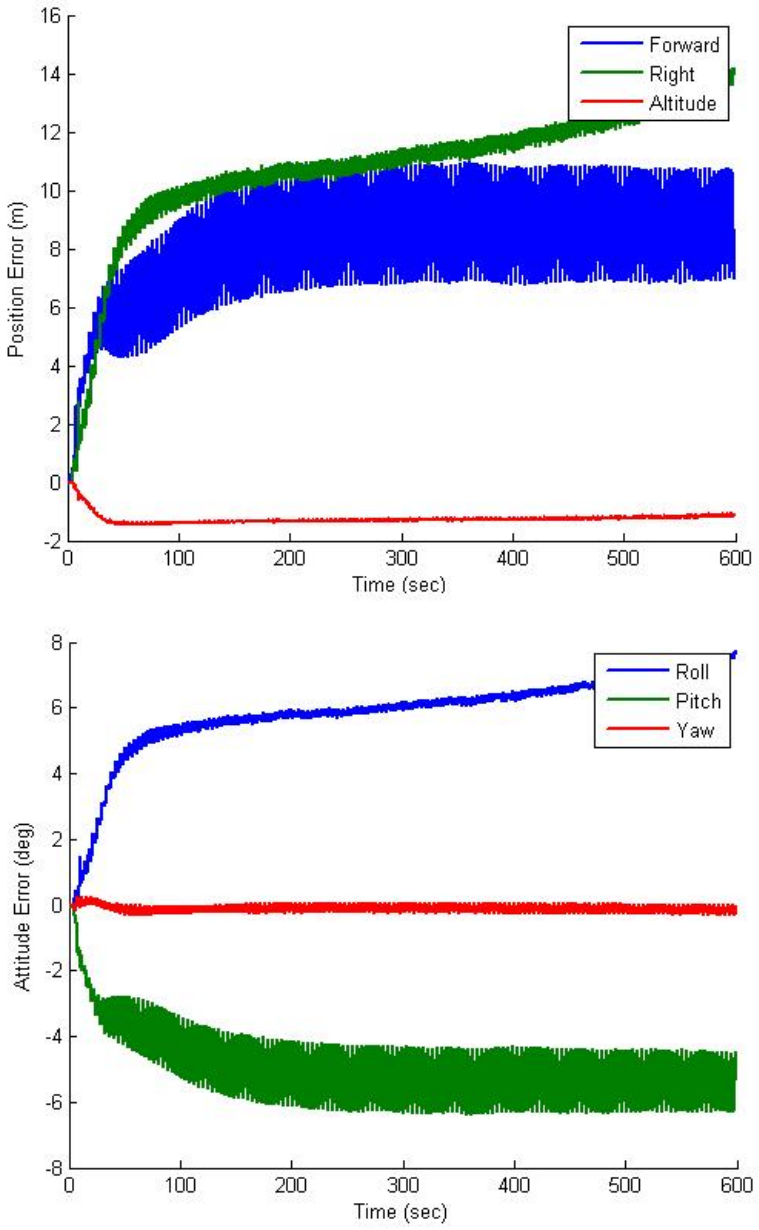


Figure 3-17: Estimation error during 10 minute simulation - single camera

Note that the error does not continue to grow with time, however there is a significant error in both position and attitude. This is caused by the confusion between translation and rotation. Despite the sufficiently disperse features, the small motion and the uncertainty in depth combine to create an ambiguity. The best way to counteract this ambiguity is to add measurements from another camera (in this case one pointing to the right). The error results presented in Figure 3-18 demonstrate the

gains achieved by utilizing multiple cameras.

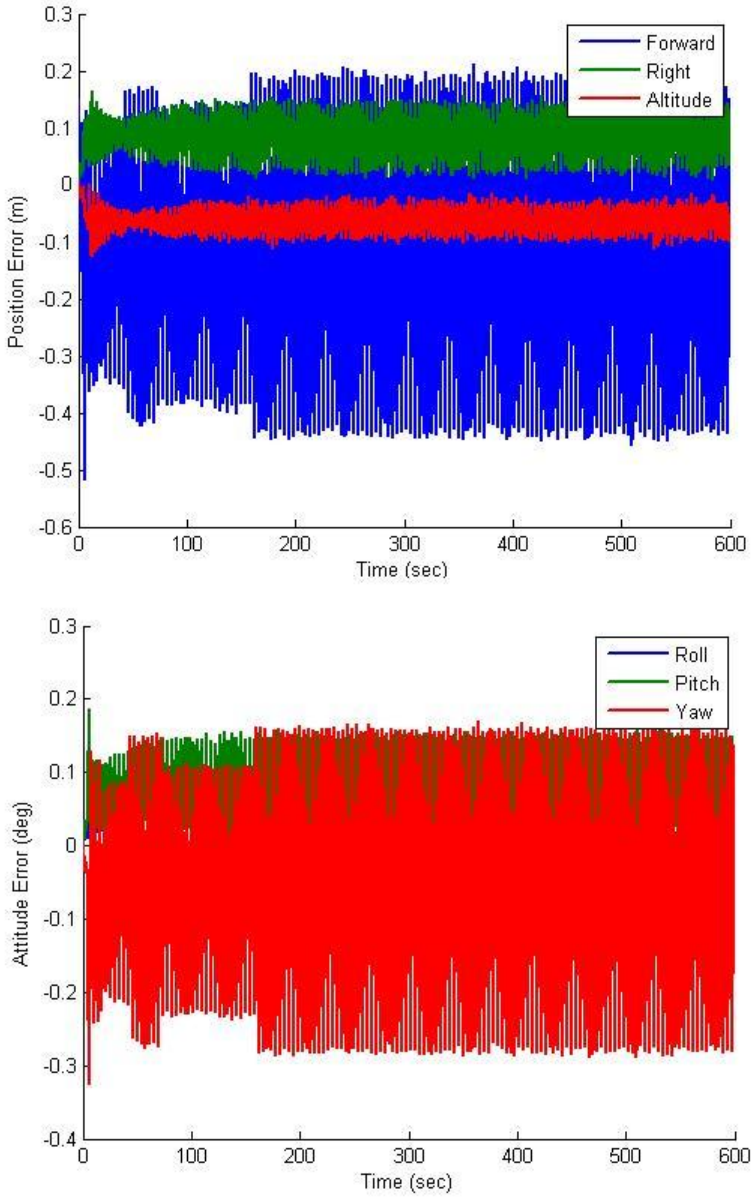


Figure 3-18: Estimation error during 10 minute simulation - two cameras

Note that in this simulation all position errors are less than 50 centimeters and rotation errors are less than a third of a degree. This is a strong demonstration that for a UAV loitering in the same region the vision based motion estimate will not accumulate drift with time. However, in practice it may not be feasible to use a side looking camera - it may not have enough features in the field of view to provide a reasonable measurement. In this case it may be useful to use two downward looking

cameras separated by a certain angle. If the camera orientations are sufficiently different, and the features in the two views are distinct, similar accuracy may be obtained with two downward looking cameras. The error results presented in Figure 3-19 are achieved using two cameras angled 30° to the left and to the right of straight down. The resulting motion estimates are at least as accurate as the case with completely orthogonal cameras.

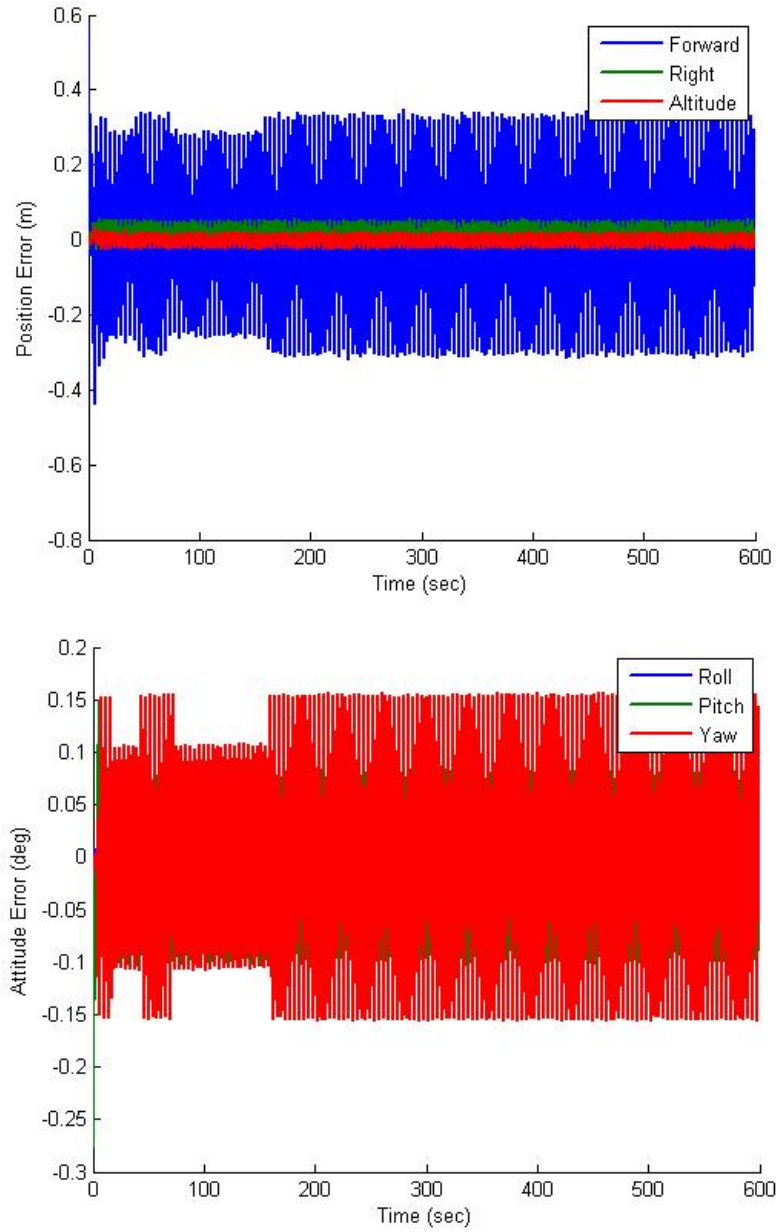


Figure 3-19: Estimation error during 10 minute simulation - two downward looking cameras

## 3.4 Limited Feature Persistence

In this section flight trajectories that produce limited feature persistence are examined. For these simulations the aircraft is maneuvered through a randomly distributed feature cloud with a maximum distance of 200 meters at a forward velocity of ten meters per second. In this scenario the maximum possible feature persistence is twelve seconds in forward flight using a downward or side looking camera, although the average feature persistence is considerably shorter. The thirty second flight trajectory consists of forward flight at a constant speed, a sweeping rolling motion, and occasional changes in altitude and lateral velocity. The true trajectory is presented in Figure 3-20 and demonstrates complex motion in all three translational directions and one rotational direction. This trajectory approximates flight through an urban canyon. The features below and to the sides of the vehicle are considerably closer than the features in front and behind the vehicle.

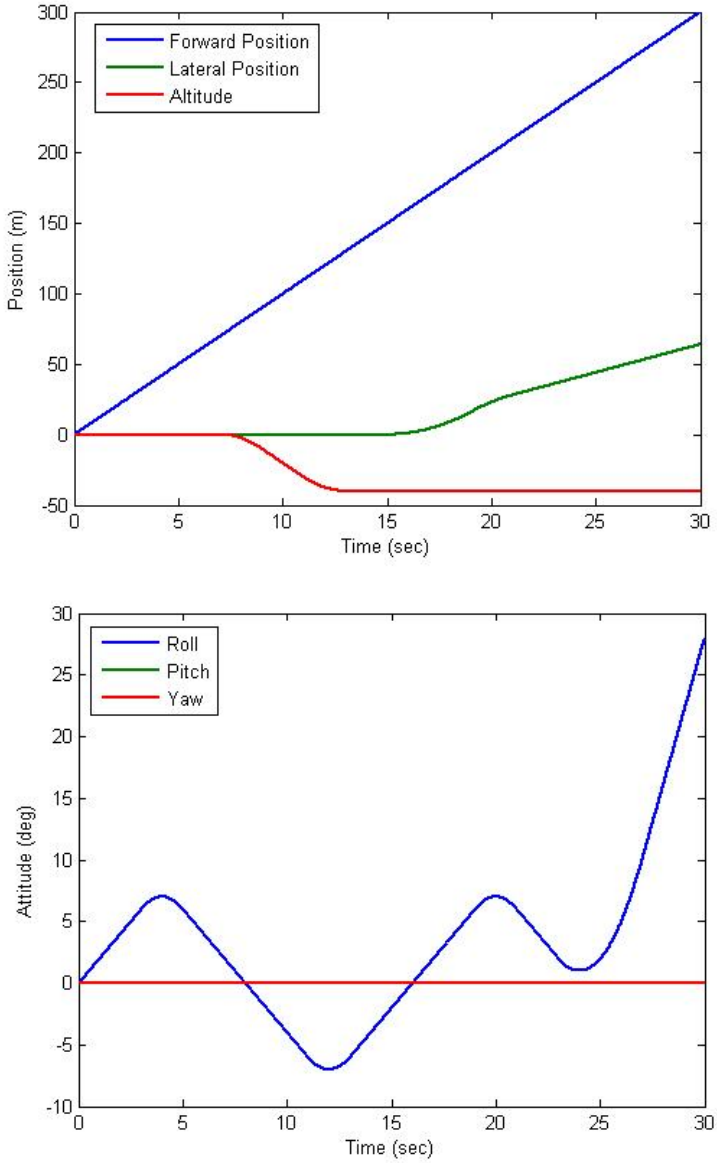


Figure 3-20: Trajectory for limited feature persistence simulation

### Single Downward-Looking Camera Results

Using a single camera oriented perpendicular to the major direction of motion (forward), the motion estimate shows some interesting properties. The position estimate (shown as an error plot in Figure 3-21) is reasonably accurate, although it shows clear trends of increasing drift in all translational directions. Recall that the maximum feature persistence is twelve seconds, and that the average feature persistence is only six seconds. As expected, as the vehicle moves into new feature areas, drift accumulates in the translational direction. Once errors in position start to accumulate, the attitude estimate is soon corrupted as well. For a trajectory in which the vehicle enters completely new regions the motion estimate is reasonably accurate for a good distance. With an average feature persistence of six seconds, this estimate is accurate while traveling through five new regions. After this period the estimate begins to dramatically diverge.

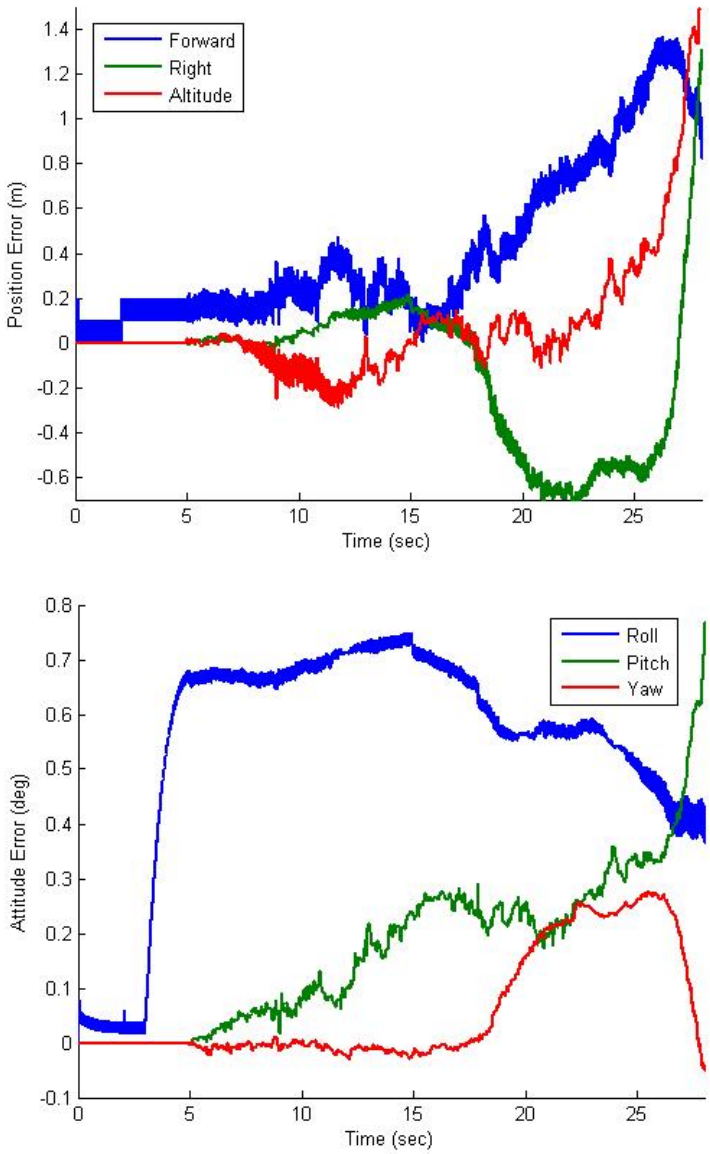


Figure 3-21: Motion estimate for limited feature persistence simulation using a single downward looking camera

### Single Rear-Looking Camera Results

Although the feature persistence for a downward or side looking camera is limited in this case, accurate estimates may be obtained by orienting a camera in a direction with increased persistence. In many cases the camera direction with the greatest feature persistence is to the rear of the aircraft. The error plot of the results for this camera configuration are shown in Figure 3-22.

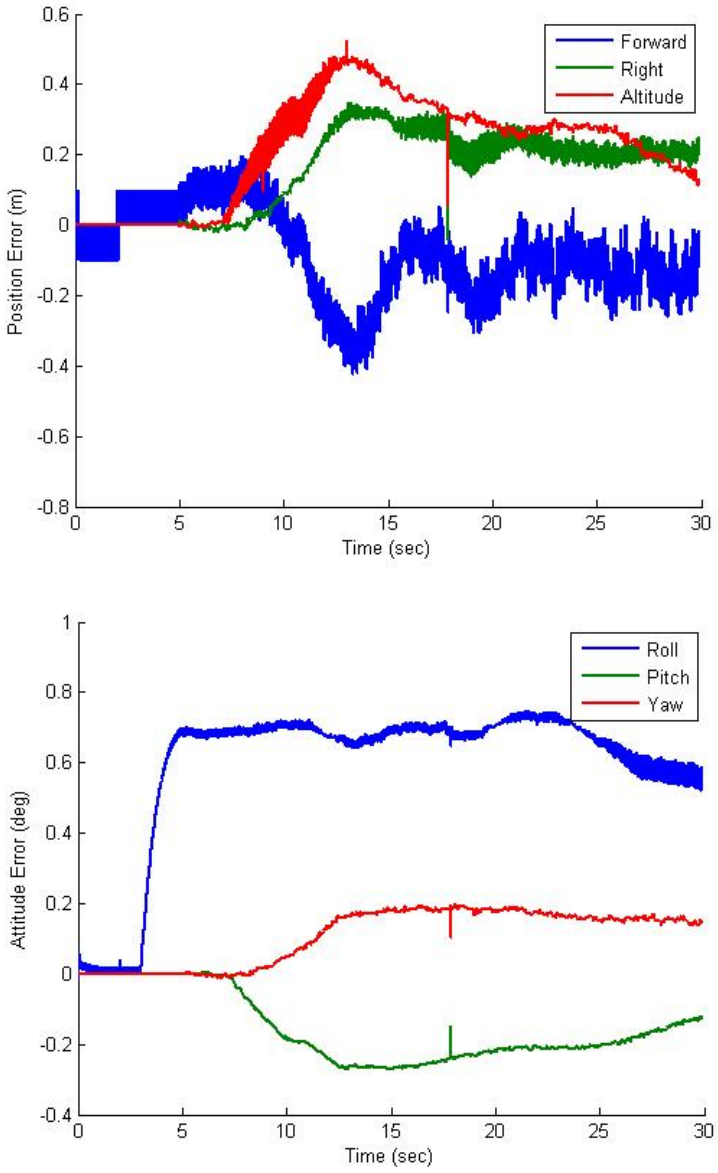


Figure 3-22: Motion estimate for limited feature persistence simulation using a single rear looking camera

Looking backward creates some difficulties when the features are very far away. If the features are sufficiently far away they will appear toward the center of the image. Forward motion will cause these features to move only a few pixels, in comparison to features that are closer to the camera and are located closer to the perimeter of the image. In many cases where a downward or side looking camera has insufficient feature persistence, a rear-facing camera will be able to track these same features for an extended period of time. Additionally, a rear-facing camera has certain advantages over a forward facing camera. The natural biological choice for orienting a camera is to have it looking forward. For this sort of situation that intuition is not always correct. A forward looking camera has the same limitations as a rear-facing camera when features are sufficiently far away and the pixel velocities are fairly small. However, it also has the limitation that when a feature is providing a great deal of information about the motion (when it is located toward the perimeter of the image), it is also about to be lost as it moves behind the camera. With a rear-facing camera good features tend to stay in the field of view for extended periods of time.

In this situation it is not absolutely necessary to integrate measurements from multiple cameras. Using both a downward looking and a rear looking camera does not significantly improve the quality of the motion estimate over the use of a single, rear looking camera. However, there is no way to know a priori in which direction the feature persistence will be the greatest. Additionally, if the direction of motion changes midway through a flight it is useful to have cameras pointing in different directions to unambiguously capture all types of motion.

### 3.4.1 Maximizing Performance

Given a certain feature persistence, the accuracy of the motion estimate can be altered by changing the length of time that features are tracked by the subfilter before being added to the main filter states. As the feature persistence decreases there is accordingly less time to track the feature in the subfilter before integrating it into the main filter. In a case where features persist for twenty seconds, it is desirable to track the feature for up to five seconds to obtain an accurate depth estimate before using

that feature for motion estimation. However, if the feature will only persist for five seconds, it obviously must be tracked by the subfilter for a shorter period of time. If the features are so fleeting that the subfilter can only track them for one second or less, it can be expected that the motion estimate will be inaccurate. In this situation the covariance on the vision update will be rather large, and hence will do little to change the motion estimate. The case of low feature persistence results in a motion estimate that is based solely on the IMU.

Initializing the filter with the correct feature depths is crucial. If the initial depths are incorrect, all subsequent feature depths will be estimated incorrectly. The initial feature depths are initialized as described in Section 2.6.5, the subfilter uses state estimates from the GPS and IMU to estimate feature depths. The subfilter uses the current state estimate to estimate the distance to new features. In general, once the depth is within 5% it is considered good enough to be place in the main filter. As the main filter proceeds it continues to update the depth of less certain features using the state estimates provided by older, more certain features. During the initialization, however, there are no existing features to aid the new features. It is recommended that the initialization be more stringent than later subfilter operation, and that initialization continue for as long as is feasible.

## 3.5 Evaluation of New Feature Subfilter

The subfilter used to estimate the initial location of new features makes use of the current motion estimate to determine the depth to each feature adding it to the main Kalman filter. No initial information is available about the depth to the feature, so that value must converge entirely within the subfilter. In this section a simulation is performed to examine the convergence of the subfilter to the correct depth values.

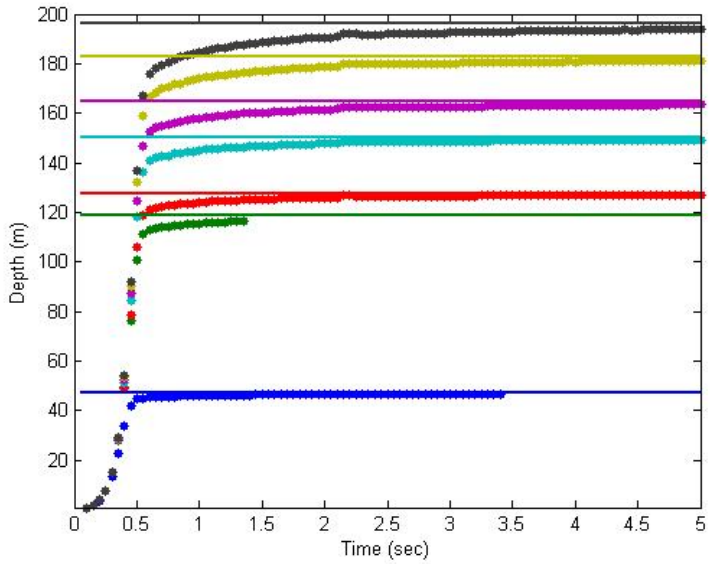


Figure 3-23: Convergence of depth estimates in subfilter

Figure 3-23 shows the convergence of the depths of seven features of various depths over a period of five seconds. For this experiment an excellent estimate of the trajectory is supplied by GPS to isolate the performance of the subfilter from errors caused by poor trajectory estimates. This data is from a case of forward translation and yaw using a downward looking camera. The horizontal lines indicate the known truth values from the feature generator (not something that is available in a real world experiment, but useful for performance evaluation in simulation). The depth estimates are represented as unconnected markers of the same color as the truth lines. For the two lowest depths the estimate ceases within five seconds. This is because those features left the field of view and were no longer available for subfilter estimation. These two features are closer to the camera and hence move across the image plane

at a faster velocity. It is clear that all depths converge to the proper value.

When integrating the subfilter data into the main filter there is a tradeoff when deciding if the feature is ready. The longer the feature is tracked by the subfilter, the better the depth estimate. However, the feature may only be in the field of view for a finite amount of time and the longer it is in the subfilter the less time it is being used by the main filter. As a starting point to determine how much time a feature should be tracked before being added to the main filter the error is examined for a number of trajectories with different maximum feature persistence. In each case the maximum error observed is recorded at various times up to half the feature persistence.

Table 3.1: Subfilter depth maximum estimation errors

| Time Tracked (sec) | 60 sec | 30 sec | 12 sec | 6 sec | 3 sec |
|---|---|---|---|---|---|
| 0.5 | 30% | 20% | 13% | 12.5% | 12% |
| 1 | 16% | 8% | 3.5% | 2.6% | 2.3% |
| 2 | 8% | 3.9% | 1.7% | 1.2% | 1.1% |
| 3 | 5.5% | 2.6% | 1.1% | 0.8% | - |
| 4 | 4.1% | 1.9% | 0.9% | - | - |
| 5 | 3.2% | 1.5% | 0.7% | - | - |
| 6 | 2.8% | 1.3% | 0.6% | - | - |
| 7 | 2.4% | 1.1% | - | - | - |
| 8 | 2.1% | 1.0% | - | - | - |
| 10 | 1.7% | 0.8% | - | - | - |
| 15 | 1.1% | 0.5% | - | - | - |
| 20 | 0.8% | - | - | - | - |
| 25 | 0.7% | - | - | - | - |
| 30 | 0.6% | - | - | - | - |

Clearly there is a dependence both on the amount of time the feature is tracked and the distance on the image plane the feature has traveled. In the limiting case imagine a feature that is stationary in the image plane, yet is tracked for a long period of time. The depth estimate for this feature should be very poor. There is a complicated relationship between the time the feature has been tracked, its motion on the image plane, and the confidence in the estimate. Fortunately, the state error covariance in the subfilter provides us with exactly the information desired in this situation - the

confidence in the feature. The covariances for one representative feature in each of the five cases presented above are plotted in Figure 3-24. These covariances have been scaled in the vertical direction by the maximum covariance observed and in the vertical direction by the feature persistence value. So at 0.5 on the x-axis each of the features has persisted for half of the maximum feature persistence. Together with Table 3.1, these plots demonstrate that in all cases the covariance is directly proportional to the depth estimation error. This is to be expected, as that is the definition of covariance in a Kalman filter. Note that the covariances for three and six second persistence do not behave as predictably as those of longer persistence. This is yet another indication that it is more desirable to have features persist for a longer period of time.

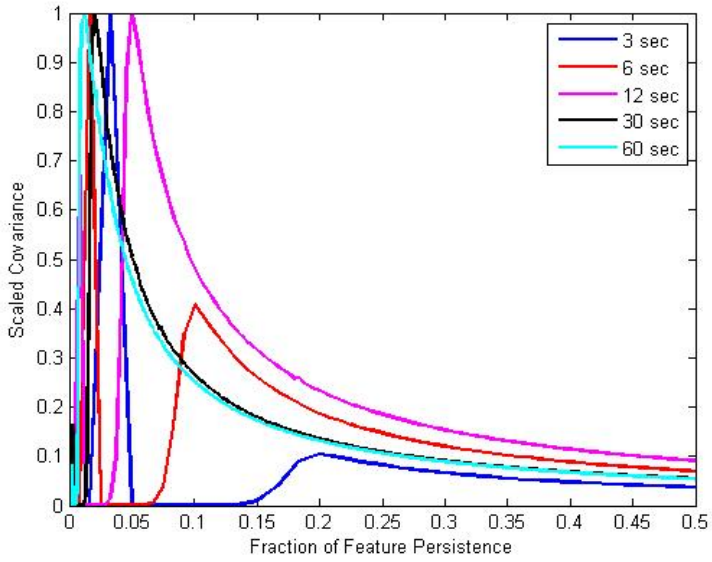


Figure 3-24: Subfilter depth covariance

Using these results, some recommendations can be made for the amount of time a feature should be tracked by the subfilter before being added to the main filter. Obviously a simple criterion of time tracked is insufficient. Simply tracking the covariance also seems troubling in the first few seconds. In this study a combination criterion was chosen in which the feature must be tracked for a certain period of time and the covariance must drop below a certain threshold.

# Chapter 4

# Conclusions

In this thesis a capability has been developed to conduct accurate UAV egomotion estimation in the absence of global positioning. Integrating feature position measurements from multiple cameras oriented in different directions and inertial measurements from an IMU, accurate position and attitude estimates are achievable. This capability has been achieved through the implementation of an extended Kalman filter that estimates the position of each feature in the camera fields of view as well as the translation and rotation undergone by the vehicle.

The Kalman filter was tested extensively in simulation to evaluate the performance in various flight regimes and the influence of using multiple cameras. The two main areas that using multiple cameras proved to be useful is in reducing the translation-rotation ambiguity and handling the problem of short feature persistence.

It was found that a significant advantage of having multiple cameras is the ability to reduce the translation-rotation ambiguity present when using a single camera. By looking in multiple directions it is found that the information obtained by the measurements is sufficient to render observable the unobservable modes of the system. Multiple cameras prove to be considerably more useful for reducing this ambiguity than a single camera with an IMU, as the small amounts of drift in an IMU cannot always decouple the motion if rates are sufficiently slow.

Perhaps the greatest value of using multiple cameras is the flexibility to fly in a number of different flight regimes without reconfiguring the hardware. For certain applications, such as high speed flight through an urban canyon, it may be demonstrated that a single camera is sufficient to capture the motion. In this case a rearward-looking camera is desired because it has the ability to track useful features for the longest period of time. However, a UAV configured with a rear-facing camera is reasonably useless in another flight regime, such as a high altitude loitering flight trajectory. If multiple cameras are configured on the vehicle and the flight dynamics change significantly, the egomotion estimator is more likely to be robust to this change than a single camera configuration.

An interesting result of using a vision sensor for motion estimation is that, unlike estimation using an IMU, error does not accumulate with time. As long as the original features remain in the field of view and the system is observable, error does not accumulate. Rather, drift accumulates only as the camera moves into a new environment. When the original features are lost and new features are added to the filter state the uncertainties in feature depth and initial position contribute to errors in the motion estimates. It is primarily for this reason that it is desirable to orient at least one camera in a direction where the feature persistence is greatest. The fewer the number of new sets of features encountered by the camera, the lower the estimation error.

In all situations it is desirable to track an unclustered group of features in an image. Feature clustering reduces the information content of the measurement and can induce a motion ambiguity. It has been demonstrated that the information content of a measurement involving a small number of unclustered features is far superior to one with a large number of clustered features. Care should be taken at all times when selecting features to ensure adequate feature dispersion.

# Appendix A

# Extended Kalman Filter Formulation

The Kalman filter is a recursive estimator which computes an estimate of the current state using only the previous state estimate and the current measurement. The filter is a two-step process: first the previous state estimate and state error covariance matrix are propagated forward in time subject to the system dynamics; next a measurement is used to update the propagated state and state error covariance. In this discussion, the notation $x\left(t|t\right)$ is used to refer to the state estimate at time $t$, $x\left(t+1|t\right)$ refers to the propagated state estimate, and $x\left(t+1|t+1\right)$ refers to the updated state estimate at time $t+1$, An 'extended' Kalman filter refers to the fact that the system dynamics are nonlinear. At each time step the system is linearized about the current estimate to calculate the Kalman gains. As with any nonlinear system, the linearization is only valid close to the linearization point. If the state estimate is ever sufficiently far from the true state, the linearization may not be valid and the filter may diverge.

### A.0.1 State Dynamics and Measurement Models

The process being modeled is given in the generic form

$$x\left(t+1\right)=f\left(x\left(t\right)\right)+w\left(t\right) \tag{A.1}$$

where the process noise $w\left(t\right)\sim N\left(0,\Sigma_w\right)$ is a white zero-mean Gaussian noise with covariance $\Sigma_w$. The state dynamics are given by $f\left(.\right)$ and in general are assumed to be nonlinear. Additionally, there is a measurable quantity $y\left(t\right)$ that is related to $x$ through the measurement equation

$$y\left(t\right)=h\left(x\left(t\right)\right)+n\left(t\right) \tag{A.2}$$

where the measurement noise $n\left(t\right)\sim N\left(0,\Sigma_n\right)$. $h\left(.\right)$ may also be nonlinear.

It is necessary to linearize the state dynamics and the measurement equation to propagate the covariance matrix and to calculate the Kalman gains.

$$F\left(x\left(t\right)\right)\doteq F\left(t\right)=\left(\frac{\partial f}{\partial x}\right)|_{\hat{x}(t)}=\begin{bmatrix}\frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n}\\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n}\\ \vdots & \vdots & \ddots & \vdots\\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n}\end{bmatrix} \tag{A.3}$$

$$H\left(x\left(t\right)\right)\doteq H\left(t\right)=\left(\frac{\partial h}{\partial x}\right)|_{\hat{x}(t)}=\begin{bmatrix}\frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \cdots & \frac{\partial h_1}{\partial x_n}\\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \cdots & \frac{\partial h_2}{\partial x_n}\\ \vdots & \vdots & \ddots & \vdots\\ \frac{\partial h_n}{\partial x_1} & \frac{\partial h_n}{\partial x_2} & \cdots & \frac{\partial h_n}{\partial x_n}\end{bmatrix} \tag{A.4}$$

$F$ is known as the *state transformation matrix* and $H$ is the *measurement sensitivity matrix*.

It is assumed that the process noise and the measurement noise are uncorrelated

$$E\left[w\left(t\right)n\left(t\right)^T\right]=0 \tag{A.5}$$

The state estimate vector $\hat{x}(t)$ is the best estimate of the actual system state and is defined as

$$\hat{x}(t) = E[x(t)] \tag{A.6}$$

The state error is then defined as the difference between the state estimate and the actual state

$$\tilde{x}(t) = \hat{x}(t) - x(t) \tag{A.7}$$

Finally, the state error covariance is defined as

$$P(t) = E\left[\tilde{x}(t)\,\tilde{x}(t)^T\right] \tag{A.8}$$

The covariance provides a statistical measure of the uncertainty in $\hat{x}$. [6]

### A.0.2 Prediction Step

In the absence of a sensor measurement, the state estimate is propagated according to the nonlinear state dynamics given in Equation A.1. The propagation according to known state dynamics produces an unbiased state estimate at a later point in time. [6]

$$\hat{x}(t+1|t) = f(\hat{x}(t|t)) \tag{A.9}$$

The state error covariance matrix must also be propagated forward in time between measurements, and this is done using the state transformation matrix given in Equation A.3

$$P(t+1|t) = F(t)\,P(t|t)\,F^T(t) + \Sigma_w \tag{A.10}$$

The size of the random system disturbance ($\Sigma_w$) has a direct influence on the growth of the error covariance. During intervals without a measurement update, the covariance will grow due to this noise, indicating a declining confidence in the state estimate.

### A.0.3 Update Step

At time $t+1$ a new measurement becomes available from the sensors, and this measurement is used to correct the predicted states from the prediction step. The new measurement $y\left(t+1\right)$ is compared to the estimated measurement $\hat{y}\left(t+1\right)$, Equation A.11, to obtain the innovation, Equation A.12.

$$\hat{y}\left(t+1\right) = h\left(\hat{x}\left(t+1|t\right)\right) \tag{A.11}$$

$$\tilde{y}\left(t+1\right) = y\left(t+1\right) - \hat{y}\left(t+1\right) \tag{A.12}$$

The innovation, or measurement residual, represents the amount the state estimate must be corrected to accurately model the system. An innovation close to zero indicates that the state estimate accurately approximates the true state. The innovation is then multiplied by the Kalman gain to update the state estimate.

$$\hat{x}\left(t+1|t+1\right) = \hat{x}\left(t+1|t\right) + L\left(t+1\right)\tilde{y}\left(t+1\right) \tag{A.13}$$

The covariance is updated in a similar manner

$$P\left(t+1|t+1\right) = \Gamma\left(t+1\right)P\left(t+1|t\right)\Gamma^{T}\left(t+1\right) + L\left(t+1\right)\Sigma_{n}L^{T}\left(t+1\right) \tag{A.14}$$

The optimal Kalman gain, $L\left(t+1\right)$ is found through the following relations

$$\Lambda\left(t+1\right) = H\left(t+1\right)P\left(t+1|t\right)H^{T}\left(t+1\right) + \Sigma_{n} \tag{A.15}$$

$$L\left(t+1\right) = P\left(t+1|t\right)H^{T}\left(t+1\right)\Lambda^{-1}\left(t+1\right) \tag{A.16}$$

$$\Gamma\left(t+1\right) = I - L\left(t+1\right)H\left(t+1\right) \tag{A.17}$$

The Kalman gain represents the relationship between the measurement uncertainty and the state estimate uncertainty, or covariance. If the measurement uncertainty is low (high confidence in the sensors), the Kalman gain will be large. Conversely, if the covariance on the state estimate is very small, the Kalman gain will be small.

# Appendix B

# Feature Extraction From An Image Stream

To transform a video stream into data that is useful for the extended Kalman filter a set of features must be identified in each frame. The vision measurement in this filter takes the form of the pixel coordinates of a set of features in the image. Over a period of time those feature locations are compared to the location of the same features in a previous frame, and from that change in measurement the motion of the camera can be determined. It is important that the same features be tracked for as long as they stay in the field of view, to increase the accuracy of the motion estimate.

This section is presented to ensure familiarity with the basics of feature tracking. The major focus of this project has been egomotion estimation, assuming that the feature tracking has been done in a reliable manner. Although the basic feature identification algorithms have been implemented and minor modifications have been made, this area was not the dominant contribution of the thesis.

### B.0.4 Feature Identification

A great deal of work has been done in the field of computer vision to develop algorithms for extracting useful information from images. The widely accepted method for selecting desirable features in the image was developed by Jianbo Shi and Carlo

Tomasi in their paper *Good Features to Track* [18]. Shi and Tomasi propose that a desirable feature is one that is easy to track in subsequent frames. The algorithm proposed in their paper is used to locate the strongest features in a video frame by finding corners and other small, high-contrast areas with large eigenvalues [20]. This algorithm is available in the open source computer vision library *OpenCV* for use in C/C++ [2].

A sample image is shown in Figure B-1. Note that the features are shown as circles.

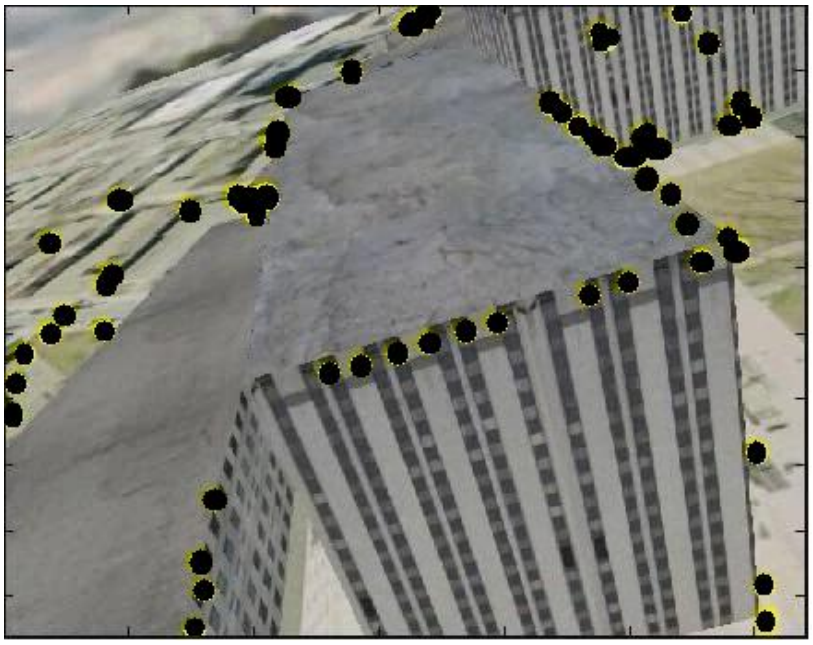

Figure B-1: Image with features found using *GoodFeaturesToTrack*

### B.0.5 Feature Tracking

To minimize the drift of ego-motion estimates, each feature is tracked through successive frames for as long as it persists in the field of view. The standard feature tracking method used in computer vision is the Lucas and Kanade optical flow technique[9]. A pyramidal implementation is available in *OpenCV* [2] and is used to track the motion of each feature found in Section B.0.4 to subsequent frames.

## B.1 Alternative Feature Tracking Methods

Standard *OpenCV* functions are used to recognize dominant features and track their location over time. A critical realization in the development of feature tracking algorithms was that in many visual environments where repetitive patterns are present (such as buildings with similar windows or roadways with repetitive markings), standard *OpenCV* algorithms need additional error checking to ensure that the features recognized in a given frame are the same as those found in previous frames. This is accomplished by predicting the location of each feature using a first order velocity calculation, and eliminating points that fall outside a bounding region.

This principle is illustrated in Figure B-2. This illustration represents the tracking of a single feature over a period of three frames. In the first frame the feature is observed at the location of the circle and in the second frame it is at the location of the triangle. If this image were a series of repeating features (windows, for example), it is possible that the region of pixels may be confused with a region at the square point. Assuming the frame rate is high enough, and the motion is not erratic, this feature tracking is likely incorrect. Logic dictates that the feature should be found somewhere in the grey bounding box. By restricting the search to the bounding box, it may be possible to detect the feature in that region.

Restricting the search area to the orange bounding box serves two purposes. First, it increases the accuracy of the feature tracking by eliminating gross outliers. Second, it reduces the computational intensity of the feature tracking process. Instead of searching for a feature in the entire image, only a small region must be searched.

For more accurate feature tracking, egomotion estimates could be used to create a better estimate of the expected feature position; this is what we call a tight coupling implementation. This could provide higher accuracy in the feature tracking, although the overall gain may be marginal. Only the first order forward-difference method has been implemented thus far, and it appears to work well. In a situation where the frame rate is high enough ($\Delta t$ is small), the higher order terms governing the motion will tend to zero.

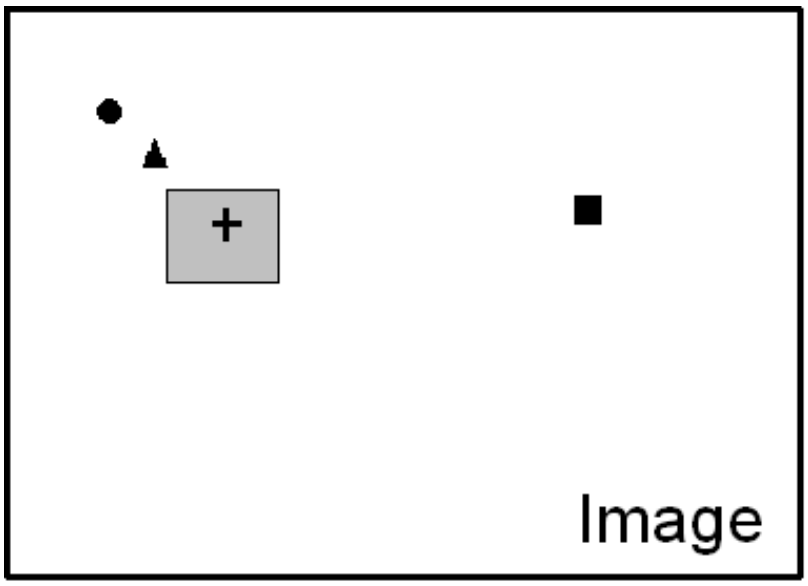


Figure B-2: Feature tracking correction

Two different feature tracking methods have been implemented, which are described in the following sections.

### B.1.1 Large Number Of Features

One method for feature tracking is to look for a large number of dominant features. These features are allowed to be anywhere on the image and are chosen to be the most dominant features in the image. The screenshot shown in Figure B-1 is an example of this feature tracking method. This method has been implemented by maintaining four bins, currently carrying 10 to 100 features each. These bins are filled at staggered intervals to ensure that many tracks are continuously maintained. When the number of features persisting in a given bin drops below a threshold (say twenty features), the entire bin is refreshed with new features from the current frame. Additionally, only one of the bins is allowed to refresh in a single frame. This maintains sufficient features at all times to determine vehicle motion. Estimates of feature quality are based in part on the number of frames that a feature has persisted. Quality estimates are passed to the ego-motion estimator such that a feature that has persisted for many frames is given more weight than a feature that has only persisted for two frames.

#### Advantages

This method is fairly easy to implement. A single call of the Lucas-Kanade feature tracking algorithm for each bin of features is sufficient. Additionally, no masking of the image is necessary - the feature tracking occurs over the entire image.

#### Disadvantages

In practice the dominant features tend to cluster in the image plane. If there is one object in the corner of the image that is substantially more distinct than the rest of the image a majority of the features may cluster around this object. While in general more measurements results in a greater confidence, having multiple features representing virtually the same object does not provide any additional information than a single feature. Additionally, as the number of tracked features increases the entire algorithm becomes more computationally intensive.

### B.1.2 Spatially Diverse Features

To increase overall efficiency, a scheme based on a partitioned image has been implemented. The image is partitioned into a number of segments, as represented by the red lines in Figure B-3, and a small number of features is tracked in each segment. Note that in each segment there are no more than two features tracked (a maximum of 32 total features) and that in some segments there are no features tracked. This method does not impose the requirement that a feature must exist in each segment - that would result in lower quality features.

#### Advantages

This implementation ensures that the features are identified across the entire image plane, rather than being allowed to cluster in a single region of the image. Additionally, this method results in reduced computation time by decreasing the total number of features that must be tracked. An important point to note is that the information obtained using the method described in Section B.1.1 is not lost using this method.

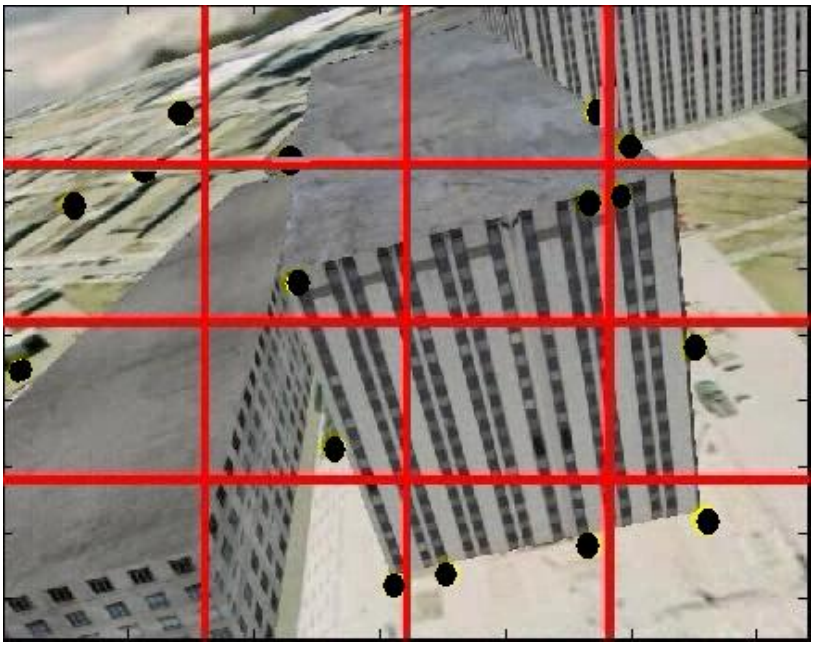

Figure B-3: Spatially diverse features

Instead of clustering multiple points on a single object, the object is represented by a single feature. This method is considered more desirable than the one previously described.

### Disadvantages

The major disadvantage of this method comes from the fact that image must be masked into segments to perform the feature tracking. And as the number of segments increases, the number of function calls to the Lucas-Kanade feature tracker also increases. This is a fairly minor disadvantage compared to the benefit of tracking a much smaller number of features.

# Bibliography


[1] A. Chiuso, R. Brockett, and S. Soatto. Structure and motion causally integrated over time. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004.

[2] Intel Corporation. Open source computer vision library. PDF, 2001.

[3] Andrew J. Davison, Yolanda Gonzalez Cid, and Nobuyuki Kita. Real-time 3d slam with wide-angle vision. *Proceedings of 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles*, 2004.

[4] J. Diebel, K. Reutersward, S. Thrun, J. Davis, and R. Gupta. Simultaneous localization and mapping with active stereo vision. *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[5] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localisation and mapping (slam): Part 1 the essential algorithms.

[6] Arthur Gelb, editor. *Applied Optimal Estimation*. The MIT Press, Cambridge, 1974.

[7] Stephen C. Paschall II. Mars entry navigation performance analysis using monte carlo techniques. Master's thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, 2004.

[8] Hailin Jin, Paolo Favaro, and Stefano Soatto. A semi-direct approach to structure from motion. *The Visual Computer*, 19:377–394, 2003.

[9] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. *Proceedings of Image Understanding Workshop*, pages 121–130, 1981.

[10] Yi Ma, Stefano Soatto, Jana Kosecka, and S. Shankar Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer, New York, first edition, 2004.

[11] Mark Maimone, Yang Cheng, and Larry Matthies. Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics*, 24(3):169–186, 2007.

[12] Milan Mandic and Emilio Frazzoli. Efficient sensor coverage for acoustic localization. *Proceedings of the 46th IEEE Conference on Decision and Control*, 2007.

[13] Stephen Maybank. *Theory of Reconstruction from Image Motion*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1992.

[14] Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, New York, 1994.

[15] Jan Neumann, Cornelia Fermuller, Yiannis Aloimonos, and Vladimir Brajovic. Compound eye sensor for 3d ego motion estimation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '04)*. IEEE, April 2004. submitted.

[16] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. *Proceedings of 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recovnition.*, 2004.

[17] L.P. Rahimi A., Morency and T. Darrell. Reducing drift in parametric motion tracking. *Proceedings of IEEE International Conference on Computer Vision*, pages 315–322, 2001.

[18] Jianbo Shi and Carlo Tomasi. Good features to track. *IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[19] James H. Taylor. The cramer-rao estimation error lower bound computation for deterministic nonlinear systems. *IEEE Transaction on Decision and Control*, 17:1178–1181, 1978.

[20] Olivier Toupet, James D. Paduano, Robert Panish, Raymond Sedwick, and Emilio Frazzoli. Augmenting state estimates with multiple camera visual measurements. *AIAA Infotech Conference*, 2007.

[21] Michael J. Veth Major USAF. *Fusion of Imaging and Inertial Sensors for Navigation*. PhD thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, 2006.